Universidade Nova de Lisboa

Faculdade de Ciências e Tecnologia
Departamento de informática

Software de sistema para um intercomunicador de veículos militares

João Alexandre de Sá Duarte

Dissertação apresentada na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa para obtenção do grau de Mestre em Engenharia Informática

Orientador: Prof. Pedro Medeiros

Monte da Caparica

2008

Para meus pais, Celina, Maria, Matilde e Marta

Agradecimentos

Gostaria de agradecer ao meu orientador Professor Pedro Medeiros pelo apoio e disponibilidade que sempre manifestou ao longo deste trabalho.

Ao Eng. Mário Guerra, director da minha unidade na EID, pela sua disponibilidade e a poio na realização desta tese.

Ao Eng. José Militão pela forma como sempre me ajudou e apoiou, tanto na realização deste trabalho, como de muitos outros realizados dentro da EID.

Aos meus colegas de trabalho, Bento, Cidália, Valente e Leonel pela ajuda e amizade.

Ao Orlando o apoio que sempre me deu, em especial nesta última fase da tese.

À minha família, em especial à minha mulher pelo apoio que sempre me deu na realização dos meus projectos.

Resumo

O projecto de Intercomunicador para Carros de Combate, designado por ICC-201, surgiu com a finalidade de substituir o antigo sistema, actualmente em uso pelo exército português, o ICC-101. A decisão de avançar para uma nova versão dos ICCs surgiu, em parte, devido à aquisição de Viaturas Blindadas de Rodas (VBR) por parte do exército português e também pela necessidade de novas funcionalidades nos ICCs. A transmissão de dados, a interligação com redes rádio PRC-525, em modo cifra ou claro, a integração com redes externas e também a necessidade de aumentar a capacidade dos ICCs e de os tornar mais flexíveis, são algumas das funcionalidades desejadas.

Esta tese vai centrar-se sobre o software desenvolvido para dois subsistemas dos ICCs, o Encaminhador e o EPC que foi realizado pelo autor desta tese. Esse software é construído sobre um sistema operativo de tempo real (Windows CE). Em particular, foi desenvolvido um conjunto de *device drivers* que permitem integrar no sistema um conjunto de periféricos não-standard. Foi também integrado software de routing que permite a interligação de vários ICC e ainda o suporte de voz sobre IP.

O software desenvolvido encontra-se integrado num produto que está instalado nos referidos VBRs do Exército Português, tendo ultrapassado um conjunto diversificado de testes.

.

Abstract

The intercom system for armoured vehicle project, known as ICC-201, started with the aim of replacing the old intercom system, currently in use by the Portuguese army, the ICC-101. The decision to move to a newer version of the ICCS arose, in part because of the acquisition of armoured wheeled vehicles by the Portuguese army and also because of the need for new features in the ICCs. Data transmission, interconnection with the PRC-525 radio networks and the need to increase the capability of the ICCS and make them more flexible, are some of the features desired.

This thesis will focus on the software developed for two subsystems of the ICCs, Encaminhador and EPC that has been developed by this thesis´s author. This software runs over a realtime operating system (Windows CE) and includes a set of device drivers that integrates several non-standard devices in the operating system. The work also included the porting of software supporting the interconnection of several ICCs and of voice over IP facilities.

The software developed is installed in a commercial product that is currently installed in armoured wheeled vehicles of the Portuguese Army. Before the deployment in the client, the system and its software have passed a demanding set of tests.

Conteúdo

Lista de Figuras

1

Introdução

Com esta tese pretende-se descrever o trabalho realizado pelo autor no desenvolvimento de um intercomunicador para carros de combate na Empresa de Investigação e Desenvolvimento (EID).

Um intercomunicador de carros de combate tem essencialmente como função, assegurar as comunicações dentro do veículo, entre os seus tripulantes e também passageiros. O objectivo é assegurar que todos consigam comunicar entre si, num ambiente normalmente caracterizado pelo ruído excessivo. Para além da comunicação entre tripulantes e passageiros, os tripulantes devem também conseguir receber informação sobre o veículo, como por exemplo o painel de alarmes, o sistema de controlo do armamento, entre outros. Para além de ter de assegurar as comunicações com o interior, um intercomunicador deverá ainda assegurar as comunicações entre o interior do carro de combate e o exterior.

Por muito bom que seja um carro de combate, se os seus tripulantes não conseguirem comunicar entre si e com o exterior, porque o barulho proveniente do movimento do veículo ou do sistema de armamento se sobrepõem às suas comunicações, o carro de combate poderá ser considerado inútil.

O projecto a que esta tese se refere foi identificado como Intercomunicador para Carros de Combate (ICC) e designado por P/ICC-201. Este projecto surgiu com a

finalidade de substituir o antigo intercomunicador, actualmente em uso pelo exército português, o P/ICC-101. Ambos os intercomunicadores foram desenvolvidos pela EID.

Figura 1.1 - Imagem do ICC-101

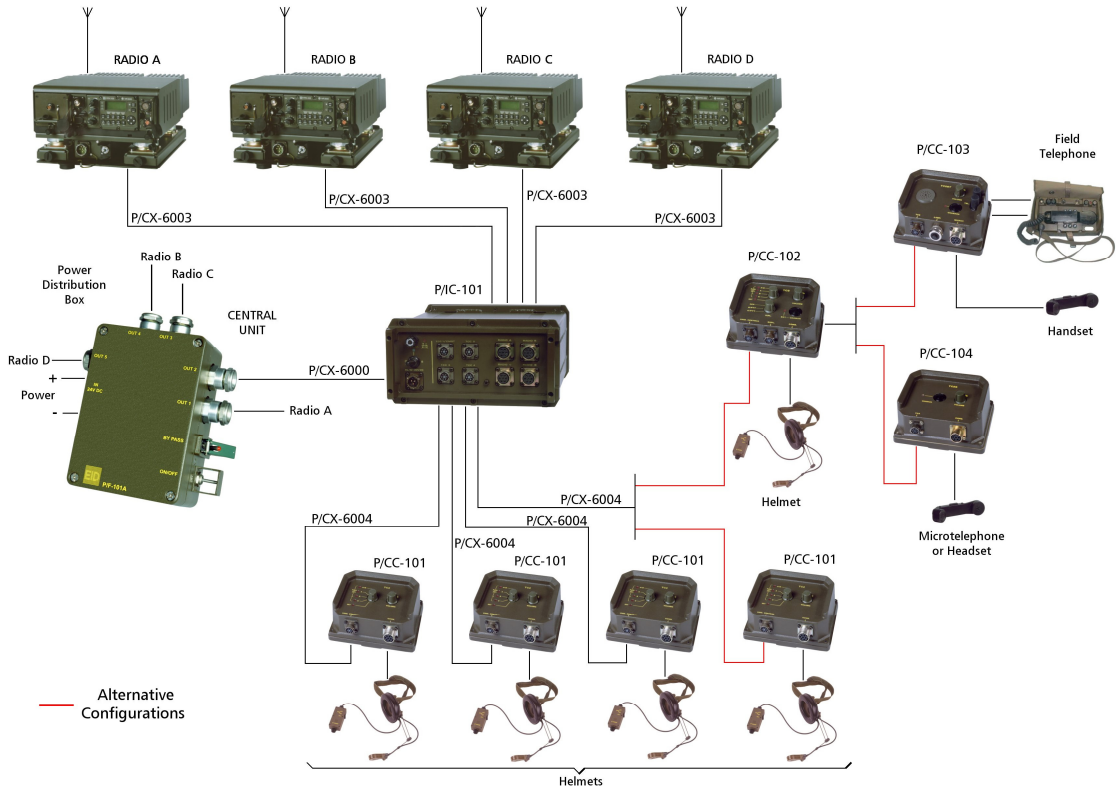


Figura 1.2 - O ICC-101 e os seus possíveis periféricos

A decisão de avançar para uma nova versão dos ICCs surgiu, em parte, devido à aquisição de Viaturas Blindadas de Rodas (VBR) por parte do exército português e também pela necessidade de novas funcionalidades nos ICCS.

Figura 1.3 - Pandur II – VBR adquirida pela exército português

Os requisitos e soluções técnicas estudadas para esta nova geração de Intercomunicadores de Carros de Combate tiveram em conta três aspectos fundamentais:

- As características da actual geração de ICCs (P/ICC-101)
- As características dos sistemas concorrentes
- A aquisição por parte do exército português das VBR - Viaturas Blindadas de Rodas para Portugal

De modo a assegurar que um equipamento militar possui as devidas características, a sua concepção e desenvolvimento deve ser feita de acordo com determinadas normas. Estas normas, conhecidas como normas de defesa, são normalmente chamadas de MIL-STD ou MIL-SPEC [1] e têm como objectivo alcançar a normalização entre os equipamentos de defesa. Para além das entidades militares, podem também ser usados por empresas ou outras organizações.

Basicamente uma norma de defesa é um documento que estabelece os requerimentos técnicos e de engenharia para processos militares.

As normas de defesa surgiram da necessidade de assegurar o perfeito funcionamento de equipamento militar. Na segunda guerra mundial os parafusos e porcas americanos não encaixavam no equipamento britânico, as normas de defesa surgiram para evitar que este tipo de problemas se repetisse no futuro. As normas de defesa proporcionam vários benefícios, tais como assegurar a compatibilidade de ferramentas e qualidade durante a produção de equipamento militar.

É possível encontrar várias referências a estes standards nos requisitos técnicos do projecto.

1.1 Requisitos gerais do projecto

Para além dos requisitos básicos para um intercomunicador, existem ainda outros requisitos relacionados com equipamento específico que se pretende utilizar, em conjunto com o intercomunicador, como por exemplo o rádio PRC-525 [2] e outros requisitos considerados essenciais para este tipo de equipamentos nos dias que correm, como seja a possibilidade de transmissão não apenas de voz mas também de dados.

Figura 1.4 - Foto do rádio PRC-525, fabricado pela EID

Figura 1.5 - O rádio PRC-525 em numa montagem veicular dupla

O anterior sistema de intercomunicadores apenas dispunha de serviço de voz, tanto para comunicações com o interior, como para comunicações com o exterior. Além disso o antigo sistema era completamente analógico, enquanto o novo sistema deve ser um sistema digital. Esta diferença de conceito permitirá que o novo sistema possua outro tipo de características, como por exemplo uma melhor imunidade ao ruído.

As exigências nas comunicações de hoje em dia não são as mesmas que existiam na altura em que se produziu a primeira versão dos ICCs. Para além da qualidade de voz exigida num sistema deste tipo e também da sua imunidade ao ruído, tendo em conta o tipo de veículo ao qual se encontra destinado, começam também a surgir outras necessidades, que acabam muitas vezes por deixar a voz para segundo plano.

A transmissão de dados é uma das exigências dos sistemas de comunicação actuais. Tal como se pode verificar no mundo civil, as comunicações de dados começam a ter, muitas vezes, uma importância maior que a própria transmissão de voz. A nova geração de telemóveis, por exemplo, tende a favorecer as mensagens e conteúdo visual em vez das tradicionais chamadas por voz. Hoje em dia todos os telemóveis possuem câmaras fotográficas e é também cada vez mais frequente o envio de mensagens (SMS) e fotos através dos telemóveis. São cada vez mais comuns telemóveis com grandes ecrãs, que permitem visualizar melhor tanto imagens como filmes. Também a possibilidade de partilhar ficheiros entre utilizadores começa a ser algo comum nos telemóveis. Assim, as comunicações do novo sistema com as redes rádio devem ser optimizadas, de modo a que, juntamente com as tradicionais comunicações por voz, se possa ter facilmente acesso aos serviços de dados.

Tendo em conta que as forças armadas portuguesas têm vindo a adquirir rádios PRC-525 (também fabricados pela EID), torna-se necessário que o novo ICC consiga interagir ao máximo com esta unidade de rádio. Uma das funções a destacar dentro da interacção entre os dois sistemas é o controlo remoto, o que permite que os rádios sejam operados sem a presença de um operador, permitindo que este à distância seleccione nos rádios os modos de operação, frequências, etc.

Apesar dos vários modos de operação possíveis para o ICC, a sua configuração deve ser simples e flexível. A configuração deve ser realizada a partir de um PC, para o qual será realizado software aplicacional que permita a configuração via Ethernet.

Como já foi referido, uma das principais características dos carros de combate é o seu elevado ruído no interior do veículo. Isto faz com que um dos requisitos naturais, seja a presença de uma funcionalidade que permita a redução de ruído, de modo a baixar para valores aceitáveis, o ruído presente dentro deste tipo de veículos militares.

É também importante que os operadores consigam comunicar mantendo ao mesmo tempo as mãos livres para outras funções. Este tipo de funcionalidade, é designada por VOX. O seu funcionamento consiste em detectar a presença de voz, que uma vez presente inicia uma comunicação, fazendo com que se torne desnecessário o habitual pressionar de um botão para iniciar as comunicações (PTT – Press To Talk).

Mesmo tendo em conta que um sistema militar deve possuir uma fiabilidade e robustez elevada, é preciso ter em conta o processo de manutenção. A manutenção do

sistema deve ser simples, de modo a que operações como retirar ou colocar um ICC numa viatura sejam realizadas com rapidez e sem grandes exigências mecânicas.

Outra das necessidades relevantes neste tipo de sistemas, é a tentativa de não ver o ICC e o veículo correspondente, como uma unidade isolada, mas sim tentar integrá-la numa rede que permita a circulação de dados entre as várias viaturas de um modo rápido e fácil de alcançar.

O facto de poder fornecer um maior número de serviços, leva a que o número de equipamentos possíveis de se ligar ao ICC seja também superior. Para além dos habituais terminais dos tripulantes, o sistema terá de estar preparado para ter ligado outro tipo de equipamentos, como por exemplo computadores portáteis.

Para evitar que o sistema tenha um tamanho demasiado grande e com demasiadas fichas, torna-se necessário jogar com as configurações do sistema, de modo a permitir que a uma mesma ficha possam ser ligados mais do que um tipo de equipamento, tornando assim o sistema flexível e fácil de operar.

Figura 1.6 - Imagem do ICC-201

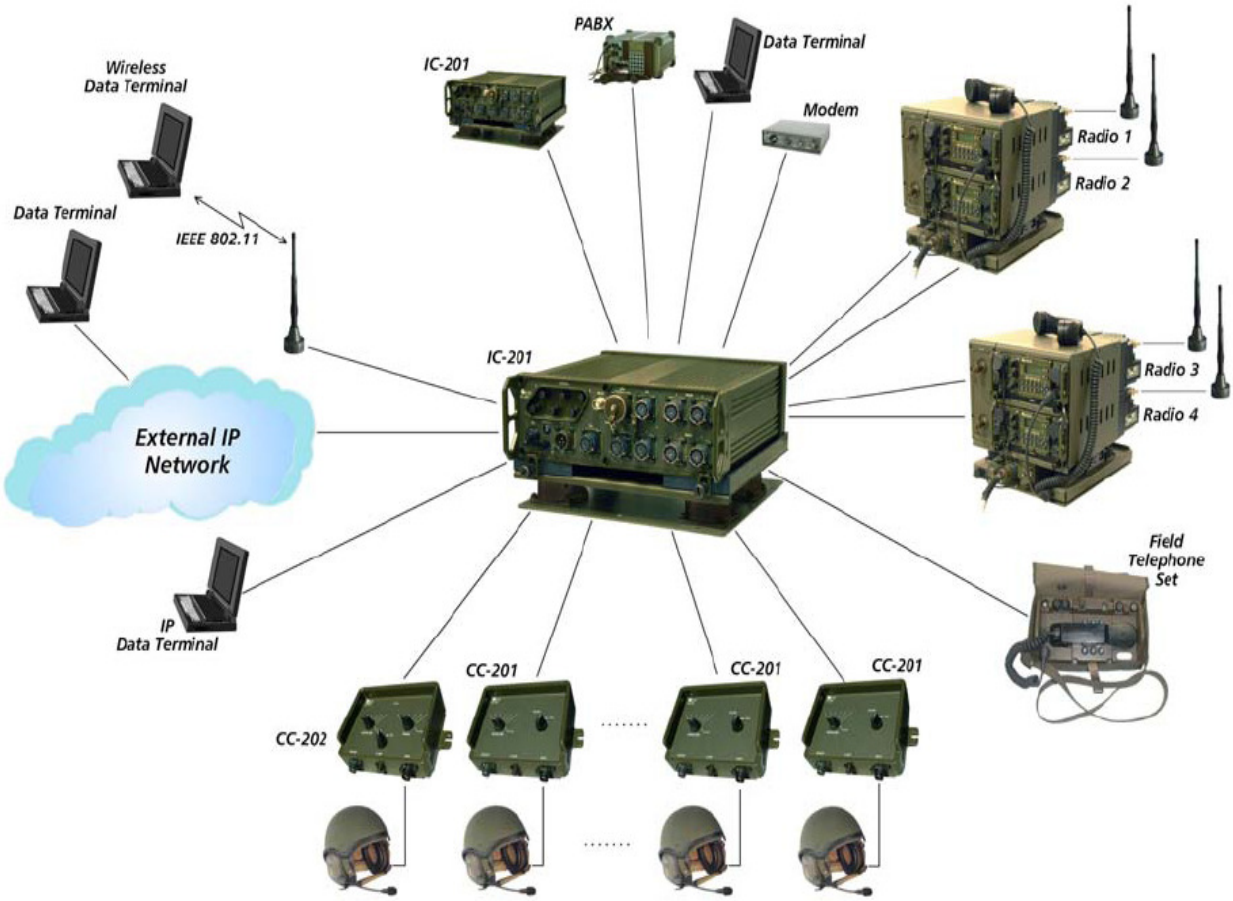


Figura 1.7 - O ICC-201 e os seus possíveis periféricos

1.2 Trabalho a desenvolver

Como já foi referido, esta tese refere-se apenas a uma parte do projecto Intercomunicador de Carros de Combate. A parte do projecto aqui descrita está relacionada com os dois sub-sistemas que devem operar dentro do intercomunicador.

Descreve-se por um lado os requisitos para esses sub-sistemas e por outro as soluções encontradas para conseguir alcançar as funcionalidades pretendidas. Os dois sub-sistemas serão sistemas embebidos, sendo o primeiro sub-sistema designado por Encaminhador e o segundo por EPC (Embedded PC).

Um dos requisitos do sistema é que as várias viaturas se consigam ligar entre si, de modo a formar uma rede onde possa circular informação entre as várias viaturas. Caberá ao Encaminhador as funções de assegurar a ligação ethernet entre as várias viaturas ou com outros dispositivos, como por exemplo portáteis. Esta ligação ethernet entre as várias viaturas permitirá que circulem dados e voz (VOIP) em toda a rede. A ligação a outros dispositivos, como por exemplo os portáteis, terá como objectivo operações de configuração, testes ou a colocação de dados na rede. Devem estar disponíveis ligações por cabo e fibra óptica. O Encaminhador terá a responsabilidade de gerir as ligações

com os ICCs e outro dispositivos que a ele se encontrem ligados, independentemente do tipo de ligação.

Tendo em conta que podem ser ligados vários dispositivos ao encaminhador e que consoante o dispositivo, pode ser necessário alterar a configuração da porta a que este se encontra ligado, qualquer alteração à configuração que o operador pretenda efectuar, deve ser rápida e simples. Além disso pretende-se que o sistema esteja inactivo o menos tempo possível quando há mudanças de configuração.

Uma vez formada a rede entre as viaturas, terá de existir um protocolo de encaminhamento para gerir a informação que circula na respectiva rede, sendo este protocolo escolhido, basicamente, em função do tipo e tamanho de rede que se pretende formar com os ICCs.

Enquanto o Encaminhador se encontra mais ligado à rede a formar pelas viaturas, o EPC encontra-se mais orientado para a gestão do sistema. O EPC deverá saber sempre em que estado se encontra o sistema e deve saber sinalizá-lo ao operador. É o EPC que lê as configurações carregadas pelo operador, para saber quais os dispositivos que podem estar ligados aos vários conectores do intercomunicador e também para saber que dispositivos se encontram ligados em determinado momento. Qualquer input gerado por um destes dispositivos é gerido pelo EPC, assim como o output correspondente.

Deverá ser o EPC a fazer a ligação entre o sistema e as aplicações de software desenvolvidas, com os objectivos de manutenção, teste ou configuração. O EPC será também o responsável pela transmissão dos pacotes VOIP, que são posteriormente encaminhados pelo Encaminhador para o destino desejado.

Qualquer acesso por parte dos operadores ao Encaminhador ou EPC será realizado por FTP (File Transfer Protocol) e deverá ser autenticado, evitando assim que um qualquer intruso tenha acesso à configuração existente.

Tanto o Encaminhador como o EPC deverão funcionar com um sistema operativo embebido de tempo real que permita tirar partido do hardware dos dois sub-sistemas e conseguir que todas as funcionalidades pretendidas sejam alcançadas. O sistema operativo será escolhido tendo em conta factores como o suporte pós aquisição, custo de compra, facilidade de utilização, facilidade de carregamento no sistema, entre outros.

Com base nesse sistema operativo serão desenvolvidos os device drivers necessários, para trabalhar com os vários componentes de hardware que se pretendem ligar aos sub-sistemas em causa e para os quais não foram disponibilizados drivers juntamente com o mesmo sistema operativo.

O hardware a escolher para os dois sub-sistemas deve ter em conta o tipo de finalidade que se pretende, um sistema do género deve ter uma boa fiabilidade e uma manutenção simples, ou seja, devem ser evitados acessórios como ventoinhas para os CPUs, de modo a reduzir ao máximo os componentes falíveis e com necessidade de manutenção. Aspectos como a capacidade de processamento e tamanho da memória devem ser tidos em conta.

Na escolha do sistema operativo deve ser tido também em conta a compatibilidade do tipo de processador que se encontra no conjunto de hardware escolhido.

1.3 Contribuição da tese

Esta tese vai centrar-se sobre os dois subsistemas do ICC, mais propriamente sobre a sua componente de sofware.

Desta componente de software fazem parte tanto o sistema operativo, como também os drivers de cada um dos sub-sistemas.

O sistema operativo será escolhido em função de factores como o preço, hardware escolhido para os sub-sistemas, suporte, entre outros. Os device drivers serão desenvolvidos em função dos dispositivos com os quais cada sub-sistema necessitar de trabalhar e do sistema operativo escolhido.

No Encaminhador, será necessário desenvolver um device driver que controle um switch comercial, de modo a que através deste switch o ICC se consiga ligar em rede, por cabo ou fibra óptica, a outros ICCs e também a outro tipo de dispositivos, como por exemplo computadores portáteis. Deve ser possível através deste driver programar o switch, aceder a informação que este recebe da rede e enviar informação para a rede através de qualquer uma das portas do switch.

No que diz respeito ao EPC, será necessário desenvolver um conjunto de device drivers, que permitam aceder a dispositivos que contêm informação essencial ao funcionamento do sistema, como por exemplo o número de série do ICC gravado numa EEPROM. Estes device drivers devem conseguir inicializar os vários dispositivos e efectuar operações de escrita e leitura sobre esses mesmos dispositivos.

Para além disso o EPC será o responsável pela geração de pacotes VoIP que serão posteriormente direccionados pelo Encaminhador para o destino correspondente.

1.4 Organização da tese

Esta tese encontra-se dividida em seis capítulos.

O primeiro capítulo pretende dar uma noção do projecto ICCs, nomeadamente de como surgiu e quais os seus objectivos. É também feita uma breve descrição do trabalho a realizar e descrita qual a relação entre o trabalho desenvolvido ao longo desta tese e o projecto ICCs. O segundo capítulo contém um pequeno estudo sobre alguns dos temas de maior interesse para esta tese, como por exemplo routing, VoIP, entre outros. No terceiro capítulo são descritas as soluções encontradas, de modo a garantir que os objectivos propostos para este trabalho sejam alcançados, juntamente com as razões que justificam a sua escolha. O quarto capítulo contém uma descrição detalhada da implementação das soluções mencionadas no capítulo anterior, enquanto o capítulo

cinco descreve os testes realizados, juntamente com os sistemas similares ao ICC. O sexto capítulo é constituído pelas conclusões e trabalho futuro.

2

Trabalho relacionado

Este capítulo pretende focar vários temas de interesse para a realização desta tese e do trabalho relacionado. Este pequeno estudo servirá de base para algumas das escolhas realizadas ao longo da execução do projecto.

2.1 Sistemas Embebidos

Em relação aos sistemas embebidos, começa-se por fazer uma caracterização deste tipo de sistemas. Descreve-se em seguida o hardware típico e em relação ao software, faz-se uma introdução dos sistemas operativos de tempo real

2.1.1 Caracterização de um sistema embebido

Os sistemas embebidos representam uma classe de sistemas computurizados dedicados, desenvolvidos para fins específicos[3]. Muitos desses sistemas são fiáveis e predizíveis[3].

As diferenças para outros sistemas de uso mais geral, como por exemplo um computador pessoal, são grandes. Ao contrário de um sistema embebido, um

computador pessoal pode executar um variado leque de tarefas. Essas tarefas variam conforme o software que se encontra a ser processado, nesse determinado momento, pelo computador pessoal. Além disso num computador pessoal é possível processar vários tipos de software, consoante as necessidades do utilizador.

Uma vez que o sistema embebido é um sistema dedicado a tarefas específicas, este pode ser optimizado, através da redução do seu tamanho e custo ou incrementado a fiabilidade e performance. Também a nível dos periféricos ditos "normais", existem grandes diferenças, os sistemas embebidos podem não ter teclado, monitor, portas série, etc. A interface com utilizador pode não existir ou quando existe é muitas vezes mínima.

Tudo isto permite reduzir a complexidade, tamanho e custo e também incrementar a robustez dos sistemas embebidos, quando comparados com os sistemas de uso mais geral.

Os sistemas embebidos são tipicamente mais baratos que os sistemas de uso geral. Factores como a produção em grande escala e a não utilização de periféricos, fazem com que estes sistemas tenham preços substancialmente mais baixos. Também o uso de processadores mais lentos e memórias com capacidade inferior, já tendo em conta o número limitado de tarefas que estes sistemas normalmente desempenham, ajuda também a que se consiga um preço mais em baixo de produção. Estas características contribuem para que estes sistemas tenham também um baixo consumo de energia, o que acaba por ser um factor de grande importância, numa grande parte das actuais aplicações deste tipo de sistemas.

Um sistema embebido contém pelo menos um processador para realizar as operações lógicas do sistema. Muitos sistemas embebidos usam um ou mais microcontroladores, que no fundo são computadores reduzido a um chip. Um microcontrolador típico contém memória suficiente e interfaces para realizar operações simples, enquanto um processador de uso geral requer chips adicionais (ex: chips de memória) para executar determinadas funções. Embora a capacidade de processamento seja reduzida, quando comparada com a capacidade de um computador dito normal, o seu baixo custo, consumo e tamanho fazem do microcontrolador a unidade ideal de processamento de muitos sistemas embebidos.

A grande maioria dos microprocessadores que são produzidos, tem como finalidade serem utilizados em sistemas embebidos, em vez de CPUs para controlar computadores pessoais. Em 2002, foram produzidos mais de 6 biliões de microprocessadores, destes, mais de 98 % foram utilizados em sistemas embebidos.

Os microprocessadores utilizados em sistemas embebidos podem ir dos mais simples de 4 bits aos mais complexos de 128 bits.

Em oposição aos computadores pessoais, onde existem poucas arquitecturas de processadores, na sua maioria x86, nos sistemas embebidos existem inúmeras arquitecturas, entre elas ARM, X86, MIPS, PowerPC, etc.

Alguns sistemas embebidos incluem um sistema operativo, que é muitas vezes referido com um sistema operativo embebido. No entanto muitos dos sistemas são de tal

modo especializados, que toda a lógica pode ser implementada como um único programa.

Muitos dos sistemas embebidos com que lidamos no nosso dia a dia, possuem características de resposta em tempo real, ou seja, a reacção a um evento externo, terá de respeitar determinados limites de tempo (na maioria dos casos milisegundos ou microsegundos).

2.1.2 História dos sistemas embebidos

Nos primeiros anos da existência dos computadores, por volta de 1940, estes trabalhavam muitas vezes dedicados a uma única tarefa, mas o seu tamanho e preço não se comparava em nada aos actuais sistemas embebidos.

O primeiro sistema a ser reconhecido como um sistema embebido moderno, foi o sistema de orientação por computador da cápsula Apollo. Este sistema era considerado o de maior risco no projecto, pelo tipo de tecnologia que utilizava, sendo que o objectivo principal da utilização deste sistema era reduzir o tamanho e peso.

O primeiro sistema embebido produzido em massa, foi o computador de orientação Autonetics D-17, feito para o míssil Minuteman, em 1961. Em 1966, foi produzido o Minuteman II e o D-17 foi substituído por um novo computador, que foi o primeiro a usar uma quantidade considerável de circuitos integrados. Com a ajuda deste projecto foi possível reduzir o preço dos circuitos integrados de maneira considerável, permitindo que ficassem acessíveis para uso comercial.

Desde as primeiras aplicações em 1960, os sistemas embebidos têm vindo a descer de preço e além disso a sua capacidade de processamento e funcionalidades têm vindo a aumentar. O primeiro microprocessador foi o Intel 4004, que foi utilizado em calculadoras e outros pequenos sistemas, mas necessitava de memória externa e chips de suporte.

Em meados de 80, muitos dos, anteriormente, componentes externos dos sistemas foram integrados no mesmo chip do processador, resultando em circuitos integrados chamados microcontroladores, tornando possível o uso generalizado de sistemas embebidos.

2.1.3 Aplicações de sistemas embebidos

O campo de utilização deste tipo de sistema é vasto, todos os dias aparecem no mercado vários produtos que utilizam sistemas embebidos de modo inovador. Não utilizar um sistema embebido no nosso dia a dia, é uma tarefa difícil.

Como exemplos de aplicações dos sistemas embebidos temos, entre outros:

- PDAs,
- Leitores de mp3,
- Consolas de jogos,

- Leitores de DVD,
- GPS,
- Impressoras,
- Microondas,
- Máquinas de lavar,
- Veículos híbridos,
- Sistemas ABS/ESC,
- Equipamento médico.

2.1.4 Limitações dos sistemas embebidos

Os sistemas embebidos possuem características que os tornam atraentes, como por exemplo o baixo custo, tamanho e consumo, mas também possuem alguns aspectos menos positivos. Os tipicamente designados "bugs", podem ser facilmente resolvidos num computador pessoal, bastando para tal processar um software de correcção, fornecido pelo fabricante. No entanto se for detectado um "bug" num sistema embebido torna-se mais complicado resolver o problema. Por norma os sistemas embebidos são programados uma única vez pelo fabricante e o tipo de sistema torna difícil carregar um software que corrija o problema. Mesmo quando é possível carregar novo software, muitas vezes o processo é complicado demais para o utilizador "normal".

Outro dos problemas associados aos sistemas embebidos é que são normalmente utilizados em situações onde falhar não é uma opção que possa ser tomada em consideração. Um sistema que controle, por exemplo, o sistema de travagem de um carro ou o sistema de navegação de um míssil, não pode falhar em caso algum. Isto obriga a que as técnicas de programação e teste utilizadas para os sistemas ditos "normais", não possam ser utilizadas no caso de sistemas embebidos. A fiabilidade do sistema tem de estar garantida antes que este deixe a fábrica, o que significa que todos os sistemas têm de ser testados e analisados exaustivamente.

Um sistema embebido terá menos recursos quando comparado com um computador pessoal, por exemplo. A capacidade de memória e processamento num sistema embebido é limitado. Isto leva a que do ponto de vista do desenvolvimento de software, desenvolver aplicações para um sistema embebido, seja mais exigente do que para um sistema "normal".

2.1.5 Hardware em sistemas embebidos

No que diz respeito ao hardware, as diferenças entre um sistema dito normal (ex: computador pessoal) e um sistema embebido podem ser bastante grandes.

Se por um lado um sistema normal possui obrigatoriamente várias interfaces com o utilizador, o mesmo pode não acontecer (e numa grande parte das vezes não acontece) com um sistema embebido, principalmente se for um sistema embebido dedicado a uma só tarefa.

No caso de existirem interfaces com o utilizador, estas podem ir dos simples botões, leds ou displays até aos sistemas gráficos de grande complexidade, onde é

possível encontrar ecrãs semelhantes aos dos sistemas normais e por vezes com algumas funcionalidades extra, como seja a sensibilidade ao toque.

Os sistemas embebidos são desenvolvidos com base em microprocessadores ou microcontroladores[4]. Os microprocessadores são aqueles que podemos encontrar nos nossos computadores pessoais e que também são utilizados por alguns sistemas embebidos. Por outro lado, os microcontroladores possuem vários periféricos no mesmo chip, o que faz com que tenham um custo e tamanho reduzido. Em qualquer um dos casos, o tamanho final do sistema é sempre muito inferior ao de um sistema normal.

Tal como os sistemas normais, os sistemas embebidos podem ter vários periféricos, sejam para comunicação série (RS-232, RS-422, RS-485), série síncrona (I2C - Inter-Integrated Circuit, SPI - Serial Peripheral Interface , etc), bus universal série (USB), interface de rede (Ethernet), conversor analógico/digital e digital/analógico, entre outros. E também periféricos que não são tão comuns de encontrar nos sistemas normais, como por exemplo JTAG (Joint Test Action Group) ou ISP (In-System Programming) que podem ser usados para debug ou carregamento de novas versões de firmware.

Uma solução comum nos sistemas embebidos é o recurso a computadores embutidos numa board, como exemplos temos o PC/104 e PC/104+, que não são mais do que computadores fabricados com a finalidade de serem pequenos. Tipicamente um destes sistemas é composto por uma motherboard, um conversor analógico/digital e um módulo de aquisição de dados.

Figura 2.1 - Imagem de um PC/104

Outra solução comum em muitos sistemas embebidos é o SoC (System on Chip), que consiste em integrar num único chip todos os componentes de um computador. Este tipo de sistema é também conhecido por ASIC (Application-specific integrated circuit), ou seja, um circuito integrado desenhado com uma função específica.

Uma solução semelhante ao SOC, seria usar uma FPGA (Field Programmable Gate Array) e programá-la. Em relação aos ASICs, as FPGAs não podem ser usados em projectos com grau de complexidade tão elevado e gastam mais energia. Como vantagens a FPGA apresenta a hipótese de poder ser reprogramada remotamente, ou localmente, de modo a resolver possíveis bugs. Também o tempo que vai do início do projecto até à entrega do cliente é menor, uma vez que a FPGA possui uma maior componente de software.

Para armazenamento de dados é comum encontrar nos sistemas embebidos dispositivos Compact Flah ou Solid State Disk. A Compact Flah é um dispositivo de armazenamento utilizado em dispositivos electrónicos portáteis. Tipicamente utiliza memória flash, que é um tipo de memória não volátil, que pode ser apagada electricamente e reprogramada.

Figura 2.2- Imagem de uma Compact Flash

Figura 2.3 - Imagem de um Solid State Disk

O Solid State Disk é um dispositivo de armazenamento de dados que também utiliza memórias do tipo Solid State, que podem por exemplo ser memórias flash, para guardar os dados. A grande diferença em relação da Compact Flash, um SSD emula uma interface semelhante à de um disco rígido, podem mesmo substituir um disco rígido caso seja necessário. As taxas de transferência são comparáveis às de um disco rígido modesto. O seu consumo é inferior ao de um disco rígido, são mais silenciosos e mecanicamente mais resistentes, uma vez que não possuem partes móveis. A sua desvantagem reside no custo.

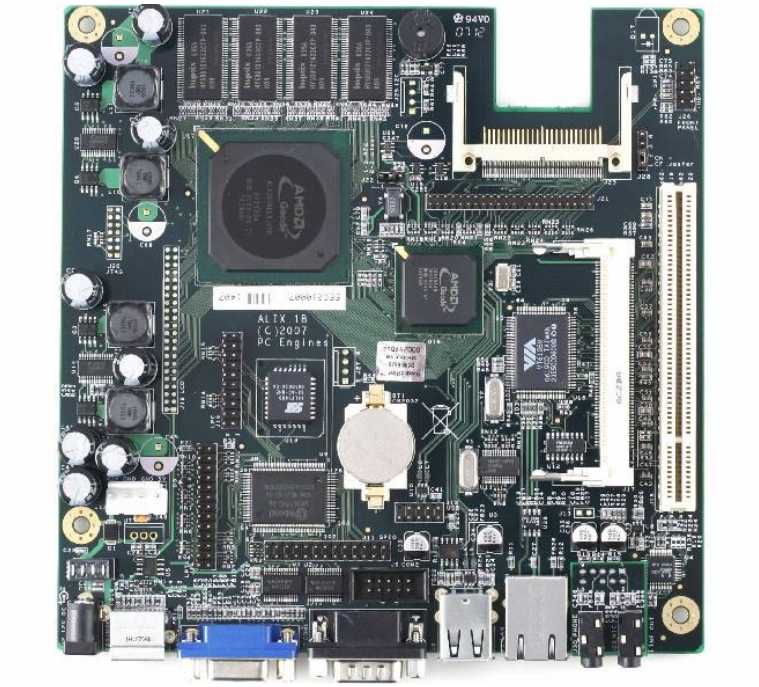


Figura 2.4 – Imagem de uma placa mini itx, muito utilizada em sistemas embebidos

2.1.6 Sistemas de tempo real

De acordo com o dicionário de Oxford, um sistema de tempo real, é qualquer sistema em que o tempo decorrido entre um input e a produção do seu output correspondente é relevante, uma vez que se não for cumprido as consequências poderão ser desastrosas. O espaço de tempo que vai do momento de input ao momento de output, deve ser garantidamente inferior aos valores especificados nos requisitos do sistema.

Esta diferença de tempo aceitável entre o momento de input e output pode variar muito de sistema para sistema, para um sistema pode ser na ordem dos microsegundos e para outro sistema na ordem dos segundos.

Os sistemas de tempo real distinguem-se dos outros sistemas exactamente nesta diferença de tempo aceitável. Num sistema de tempo real esta diferença já foi calculada, é inflexível e necessária para o correcto comportamento do sistema. Se um sistema de tempo real não responder a um evento num determinado espaço de tempo, o comportamento do sistema não pode ser considerado correcto. O mesmo não acontece noutro tipo de sistemas, onde estas diferenças de tempo não têm importância, onde o tempo que um sistema leva a responder a um evento é imprevisível e onde independentemente do tempo que um sistema leva a responder a um evento, este não se encontra a funcionar incorrectamente.

No caso de um sistema como por exemplo um computador pessoal, os tempos de resposta variam e não são considerados essenciais para o correcto funcionamento do sistema. Enquanto o tempo de resposta num computador pessoal é um factor importante, num sistema de tempo real o tempo de resposta é considerado crítico.

Para além de proporcionarem respostas a eventos externos dentro de um determinado tempo, proporcionam muitas vezes respostas múltiplas e em simultâneo.

Os sistemas de tempo real estão divididos em duas categorias: hard realtime e soft realtime.

Hard realtime

Um sistema é tempo real hard se uma falha do sistema, em responder a um evento dentro de um determinado tempo, for considerada uma falha total do sistema. Ou seja, é considerada, por aqueles que implementaram o sistema, como sendo uma falha inaceitável. Mesmo que o sistema tenha realizado milhares de funções sem falhar, uma falha apenas, continua a ser considerada inaceitável para o funcionamento previsto para o sistema. Um sistema deste tipo deve reagir a um evento sempre dentro dos limites de tempo exigidos e a todos os eventos despoletados, sem uma única falha. Ou seja, este requisito resulta do facto de o correcto funcionamento do sistema estar intimamente ligado aos tempos de resposta exigidos.

Soft realtime

Num verdadeiro sistema tempo real soft, os tempos de resposta são importantes, mas não são um caso de vida ou morte. Ou seja, um sistema continua a ter de responder aos eventos dentro de um determinado tempo, mas pode não conseguir responder a todos os eventos ou pode não responder a eventos dentro do tempo de resposta definido, sem que isso seja considerado uma falha crítica.

O número de eventos que um sistema pode falhar a resposta, é definido na implementação do sistema. O valor aceitável tanto pode ser de uma falha em 5, como uma falha em 500, ou qualquer outro valor, tudo depende de quem implementa o sistema e qual a sua finalidade.

A maioria dos sistemas de tempo real pertence à categoria dos sistemas de tempo real Soft. Os sistemas de tempo real Hard tendem a ser sistemas bastante complexos.

2.1.7 Software para sistemas embebidos

O software para sistemas embebidos é normalmente suportado por Sistemas Operativos (SO) de características diferentes do habitual, denominados Sistemas Operativos de Tempo Real.

Sistemas Operativos de Tempo Real

Um SOTR é um sistema operativo orientado para executar aplicações de tempo real [7]. Um sistema operativo de tempo real facilita a criação de sistemas de tempo real, mas para garantir que o sistema final seja mesmo de tempo real, é necessário que o software

seja desenvolvido correctamente. Um sistema operativo de tempo real não possui necessariamente uma grande capacidade de processamento, mas disponibiliza facilidades, que devidamente utilizadas, permitem obter os resultados pretendidos a nível do tempo real. Tipicamente um sistema operativo de tempo real utiliza algoritmos de escalonamento específicos, de modo a proporcionar ao programador, as ferramentas necessárias para obter um sistema final com determinado comportamento. Um sistema operativo de tempo real é valorizado mais pelo tempo que demora a responder a um determinado evento, do que propriamente pelo trabalho que consegue produzir numa determinada quantidade de tempo. Os factores chave de um sistema operativo de tempo real são a latência mínima de interupt e a latência do switching de threads.

Tal como num SO normal, num SOTR uma tarefa, ou processo, pode estar em qualquer um dos três estados seguintes: 1 – running; 2 - ready; 3 – bloqued. A maioria das tarefas encontra-se no estado blocked a maior parte do tempo. Apenas uma tarefa por CPU pode estar a ser executada.

A chave para um bom sistema operativo de tempo real reside em implementar um bom escalonador [5].

Algoritmos de escalonamento

Os SOTR utilizam normalmente uma conjugação de duas técnicas de escalonamento :

escalonamento por prioridade – é feita a comutação entre tarefas, sempre que um evento de mais alta prioridade necessite de ser atendido, isto é, a cada momento é executada a tarefa de mais alta prioridade que estava pronta para correr.

Time-sharing – a comutação de tarefas é feita com base em interrupts do clock e é usada uma estratégia de atribuição rotativa (round robin). Nas implementações por time-sharing a comutação entre tarefas é feita com uma frequência maior do que o necessário, no entanto permite fornecer uma resposta mais suave, com um multitasking mais determínistico.

Para mais detalhes consultar [6]

Comunicações entre tarefas e partilha de recursos

Os sistemas multitasking devem gerir os dados e recursos de hardware partilhados entre as várias tarefas. É considerado inseguro, ou seja, o resultado é considerado imprevisível, quando o acesso de dados ou recursos de harware partilhados por duas tarefas, é feito em simultâneo. Existem várias possibilidades para solucionar este tipo de situação, como exemplo semáforos, secções críticas, mutexes, etc [5]. Qualquer uma dessas possibilidades faz com que apenas uma tarefa consiga aceder aos vários recursos de cada vez.

Alocação de memória

A alocação de memória é mais crítica nos sistemas operativos de tempo real que nos restantes sistemas operativos.

Um factor de grande importância é a velocidade de alocação. As alocações de memória standard verificam uma lista de tamanho variável, para encontrar uma bloco de memória que se possa utilizar. Para os sistemas operativos de tempo real isto não é aceitável, uma vez que a alocação de memória num sistema operativo de tempo real tem de acontecer dentro de um determinado espaço de tempo.

É preciso ter também em conta que a memória vai ficando fragmentada, à medida que as zonas livres ficam separadas por zonas que estão em uso. Isto pode fazer com que uma aplicação deixe de funcionar, ao não conseguir obter memória, mesmo que teoricamente exista memória livre. Os algoritmos que lentamente acumulam fragmentação podem trabalhar bem numa máquina comum de uso pessoal, uma vez que quando são reiniciadas tudo volta à estaca zero, mas para um sistema embebido que pode estar anos a funcionar sem ser reiniciado, isso não é aceitável.

O algoritmo de blocos de memória de tamanho fixo é simples e funciona muito bem em sistemas embebidos. Consiste em dividir a memória em blocos de igual tamanho, que possuem uma marca que indica se estão ou não a ser utilizados. Sempre que existe a necessidade de alocar memória, os blocos de memória existentes vão sendo percorridos e a marca vai sendo consultada, os que estiverem livres são utilizados e marcados como não estando livres. Se por exemplo se pretender alocar espaço em memória para uma estrutura e essa estrutura seja maior que um bloco de memória, são alocados os vários blocos necessários e é feita uma lista dos blocos que correspondem a essa determinada estrutura, isto porque os blocos podem não estar seguidos.

Sistemas operativos de tempo real vs sistemas operativos comuns

Os sistemas de tempo real diferem dos sistemas não tempo real, no facto de terem de reagir a eventos com origem no mundo físico, dentro de uma certa quantidade de tempo[7]. Por determinista entende-se que o sistema deve consumir apenas as quantidades de tempo esperadas.

Os sistemas comuns são muitas vezes não determinísticos. Os seus serviços podem injectar atrasos aleatórios nas aplicações de software e causar uma resposta lenta por parte de uma aplicação em determinada altura.

Muitos sistemas operativos de tempo real vão para além do simples determinismo. Alguns conseguem assegurar o tempo de resposta a um evento, independentemente da quantidade de informação presente no evento ou do número de tarefas a serem executadas no momento.

Device drivers

Um dispositivo, no contexto de um sistema operativo, é um pedaço de hardware com o qual o sistema operativo tem de interagir de modo a conseguir realizar uma determinada acção. Um device driver é a entidade de software que interage directamente com o dispositivo[8]. Qualquer dispositivo, seja ele uma impressora, teclado, rato, etc, deve ter associado um device driver. Cada dispositivo possui os seus

comandos específicos, que apenas o seu driver suporta. Por outro lado, o driver aceita comandos genéricos, que traduz em comandos especializados para o dispositivo.

Assim, um device driver actua como um tradutor entre o dispositivo e o software que utiliza o dito device. Muitos device drivers já vêm incluídos no sistema operativo, como por exemplo o teclado ou o rato.

Board Support Packages

Muitos fabricantes de hardware para sistemas embebidos disponibilizam com o harware, um conjunto de software designado por BSP – Board Support Package.

Para um sistema embebido, um Board Support Pachage é o software que implementa os device drivers para um SDB (Standard Development Board)[9]. Tipicamente do BSP fazem parte um bootloader, uma camada OAL (OEM Adaptation Layer, onde OEM- Original Equipment Manufacturer) e os device drivers específicos do board.

Um bootloader não é mais que um pequeno programa que é chamado no arranque do sistema, para carregar o sistema operativo para a memória do sistema.

A camada OAL actua como interface entre o kernel e a plataforma de hardware[12]. O Bootloader depois de estar a correr, manda arrancar a camada OAL que é responsável por inicializar a plataforma, da qual faz parte, por exemplo, a inicialização dos interrupts.

2.2 Integração dos protocolos IP em sistemas embebidos

No sistema descrito nesta tese um dos periféricos presente é um controlador de rede. Desta forma o SOTR terá de controlar um periférico através de um device driver. Além disso, é suportado o conjunto de Protocolo de Internet, nomeadamente o protocolo de rede (IP), os protocolos de transporte (TCP e UDP) e alguns protocolos de aplicação relacionados com a transmissão de voz sobre IP.

2.2.1 Os Protocolos da Internet

Protocolo de rede IP

O IP (internet protocol) é um protocolo que permite o envio de dados de um dispositivo para outro na Internet. Cada dispositivo (normalmente designado por host) ligado à internet possui um endereço IP, que é único, e que o identifica na rede entre todos os dispositivos na Internet.

Quando se enviam ou recebem dados (como por exemplo um e-mail ou se visita uma página da internet) estes são considerados, do ponto de vista do dispositivo, como mensagens que terão de ser divididas naquilo que normalmente se designam por

pacotes. Cada um destes pacotes contém o endereço do originador da mensagem e o seu destinatário.

Se o destinatário da mensagem não estiver próximo, no que diz respeito a Internet, do originador, os pacotes são enviados para determinados dispositivos da rede que devem saber encaminhar o pacote até ao destino.

Uma vez que a mensagem é dividida em vários pacotes, cada pacote pode, sempre que necessário, ser enviado por um caminho diferente através da rede. Os pacotes podem chegar numa ordem diferente daquela que foram enviados. O IP apenas é responsável pela entrega dos pacotes, enquanto o protocolo TCP é responsável pelo seu reagrupamento.

O IP é um protocolo sem conexão, o que significa que não existe uma ligação estabelecida entre o originador e destinatário. Cada pacote que viaja na Internet é tratado com uma unidade de dados independente, sem que seja feita uma relação com outras unidades de dados. Cabe ao TCP, um protocolo connection oriented, relacionar os vários pacotes através do seu número de sequência na mensagem.

Protocolo de transporte TCP

O TCP (Transmission Control Protocol) é um protocolo utilizado juntamente com o IP, para a troca de dados entre dois ou mais dispositivos numa rede. Enquanto o IP é responsável pela entrega dos pacotes, cabe ao TCP manter o rastro dos vários pacotes que formam a mensagem, de modo a conseguir uma encaminhamento eficiente através da Internet.

Sempre que existem trocas de dados na Internet, as mensagens são divididas em pequenos pacotes. Apesar de cada um dos pacotes ter o mesmo endereço de destino, existe a possibilidade de percorrerem caminhos diferentes. No lado do destinatário o TCP reorganiza todos os pacotes e espera até que todos tenham chegado, de modo a ter no final, um único ficheiro.

O TCP é conhecido com um protocolo orientado para a conexão, o que significa que é estabelecida e mantida uma ligação, até que a troca de mensagens entre os dois pontos esteja acabada. O TCP é responsável por assegurar que uma mensagem é dividida nos vários pacotes, que o IP deve conseguir manusear, e também por reagrupa-los no destinatário.

Endereços e encaminhamento

Todos os dispositivos ligados a Internet possuem um identificador único que lhes é atribuído, este identificado é designado por endereço IP. Sem este endereço IP não seria possível aos dispositivos comunicar entre si.

Um endereço IP consiste em 4 dígitos separados por ‘.’, com as números a poderem variar entre 0 e 255. (Ex: 192.168.2.1)

Os endereços 0.0.0.0, 255.255.255.255 e 127.0.0.1 são reservados e não podem ser utilizados. O endereço IP deve ser único por cada dispositivo ligado à rede. Se dois dispositivos tiverem o mesmo endereço IP, existirá um conflito de IPs, o que impossibilitará os dispositivos de comunicarem entre si.

Classes de endereços IP

Os endereços IP podem ser divididos por classes. As classes possíveis para um endereço IP são A, B, C, D e E e os intervalos estão definidos na tabela seguinte.

Classes	Endereço inicial	Endereço final
A	0.0.0.0	126.255.255.255
B	128.0.0.0	191.255.255.255
C	192.0.0.0	223.255.255.255
D	224.0.0.0	239.255.255.255
E	240.0.0.0	255.255.255.255

Na tabela é possível verificar que da classe A para a B não são tidos em conta os endereços da gama 127.0.0.0 a 127.255.255.255. Esta gama de IPs encontra-se reservada.

O resto dos IPs das várias classes será distribuído por companhias e organizações com base na quantidade de endereços IP necessários.

Loopback: A gama de IPs 127.0.0.0 encontra-se reservada como loopback. Estes endereços são normalmente utilizados para testes e debug.

Broadcast : Corresponde ao endereço 255.255.255.255 e é utilizado para enviar mensagens a todos os elementos da rede, a qual pertence o dispositivo.

O endereço 0.0.0.0 é utilizado para serviços de routing.

Classe A: Disponibiliza 126 redes, que são indicadas pelo 1º dígito do endereço IP, e 16.777.214 possíveis IPs por cada rede.

Classe B: Disponibiliza 16. 384 redes , indicadas pelos 2 primeiros dígitos do endereço IP, e 65.534 endereços IP por rede.

Classe C: Disponibiliza 2.097.152 redes, indicadas pelos 3 primeiros dígitos do endereço IP, e 255 endereços IP por rede.

Classe D: Endereços reservados para serviços de Multicast.

Classe E: Endereços reservados para testes e experiências.

Redes privadas

Um rede privada é tipicamente uma rede onde os dispositivos utilizam endereços que pertencem a uma gama de endereços designada por endereços privados. Este tipo de endereços é tipicamente utilizado quando existem vários dispositivos numa rede interna, ou seja, sem estar ligada à internet ou pelo menos sem estar directamente ligada à internet.

As redes privadas são comuns nas casas dos utilizadores comuns, escritórios, etc. A sua utilização é comum porque muitas vezes não existe a necessidade de todos os dispositivos possuírem endereços únicos na rede. Isto acontece porque os IPs disponíveis com o IPv4 (endereços com 4 dígitos que variam entre 0 e 255) não são suficientes para todos os dispositivos que existem a nível mundial, o que obriga a recorrer ao uso de redes privadas.

Os dispositivos que fazem a ligação entre as redes privadas e a Internet estão programados para ignorarem todos os pacotes que contenham endereços de redes privadas, fazendo com que as redes privadas estejam isoladas entre si. O que significa que podem existir duas entidades a utilizar os mesmo endereços de redes privadas ao mesmo tempo se que existam problemas por causa disso.

Nome	Início da gama de endereços IP	Fim da Número de redes	Número de endereços
Classe A	10.0.0.0	10.255.255.255	16,777,216
Classe B	172.16.0.0	172.31.255.255	1,048,576
Classe C	192.168.0.0	192.168.255.255	65,536

Interligação de redes

Existem vários dispositivos que podem ser usados para fazer a ligação entre redes, os mais comuns são Gateway, Router, Bridge/Switch, Repetidor e sobre cada um dos quatro dá-se em seguidamente uma breve explicação[10][21].

Gateway

Um gateway é um dispositivo capaz de ligar duas redes que utilizam protocolos diferentes (Ex: IPX/SPX – TCP/IP). Um gateway pode ser implementado apenas em software, hardware ou uma combinação dos dois.

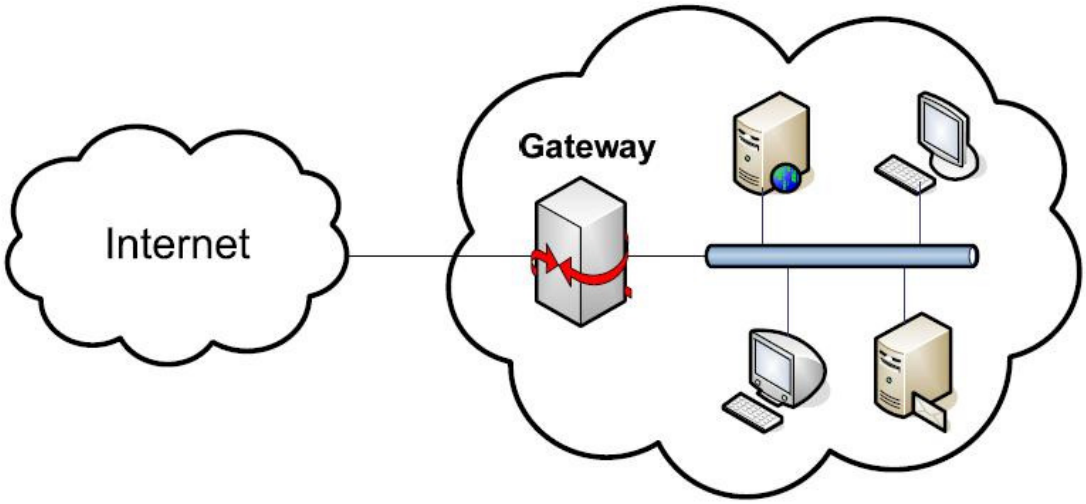


Figura 2.5 - Gateway

Router

Um router é um dispositivo que encaminha pacotes entre redes, está ligado a pelo menos duas redes. Os pacotes são encaminhados depois de determinado o seu destino e qual o melhor caminho para alcançar esse mesmo destino.

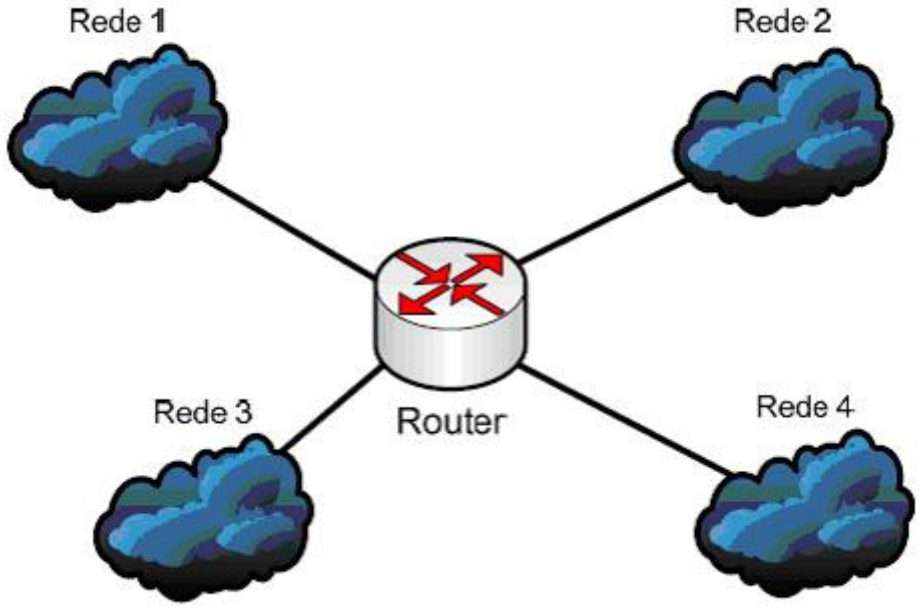


Figura 2.6 - Router

Bridge/Switch

Uma bridge/swicth é um dispositivo que liga duas redes que utilizam o mesmo protocolo (Ex: TCP/IP).

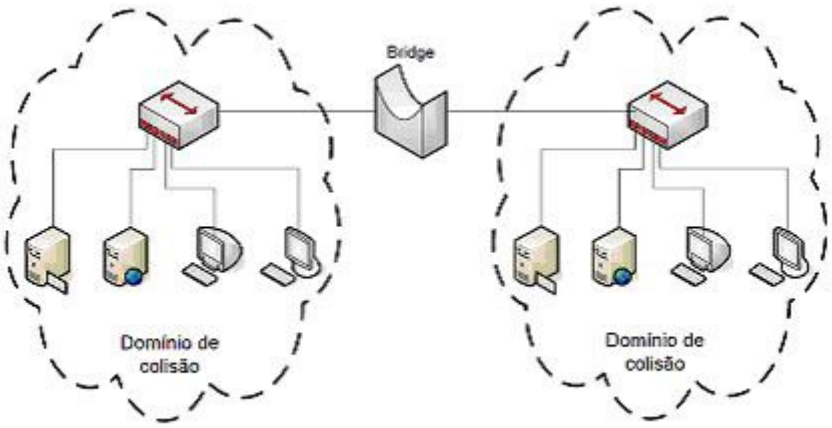


Figura 2.7 – Bridge/Switch

Repetidor

Um repetidor é um dispositivo que tem como função regenerar ou replicar sinais que estejam fracos ou distorcidos em transmissões de grandes distâncias ou em situações em que exista muita interferência.

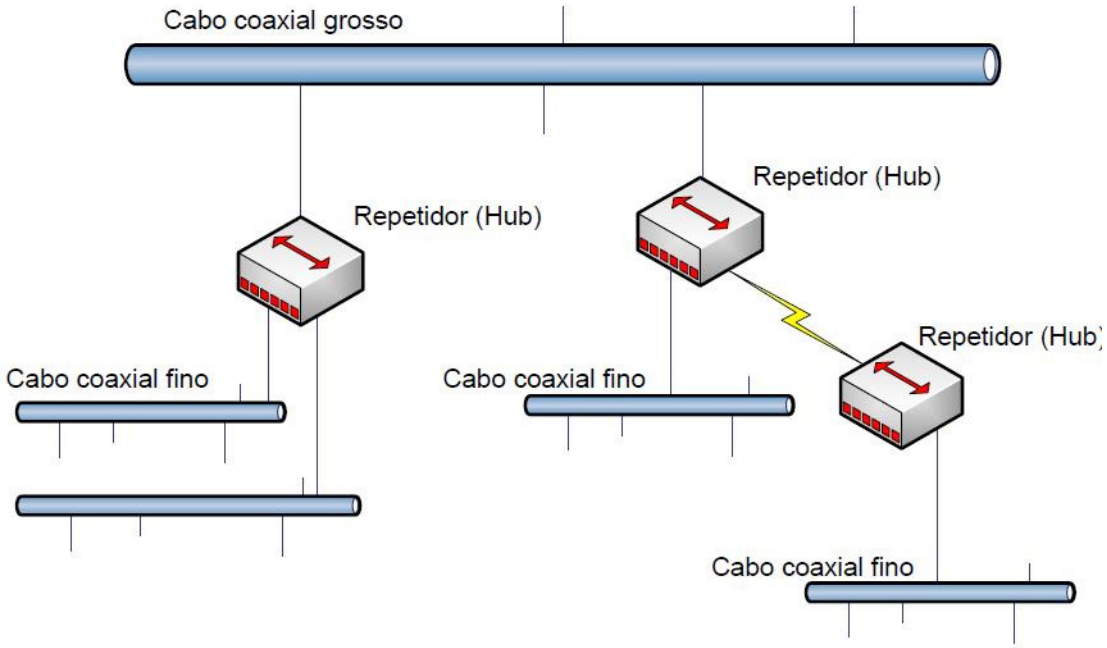


Figura 2.8 - Repetidor

2.2.2 Routing

Protocolos de routing

Pode dizer-se que network routing é a capacidade de uma rede de comunicações enviar uma unidade de infomação de um ponto A a um ponto B, determinando o caminho a percorrer na rede de um modo rápido e eficiente[13].

O trabalho de um router é determinar qual o próximo ponto da rede, para onde um pacote se deve dirigir, de modo a conseguir alcançar o seu destino. Um router encontra-se ligado a pelo menos duas redes e toma a decisão de qual o caminho para onde deve enviar um pacote, com base no estado actual das redes às quais está ligado. Um router cria e/ou mantém uma tabela dos caminhos possíveis e do seu estado. Esta informação é utilizada juntamente com os algoritmos de distância e custo, para determinar o melhor caminho que um determinado pacote deve percorrer para alcançar o seu destino. Tipicamente um pacote pode passar por uma série de routers antes de chegar ao seu destino.

Um protocolo de routing é um protocolo que especifica o modo como os routers devem comunicar entre si, de modo a difundir a informação que lhes permite seleccionar o melhor trajecto, entre dois pontos de uma rede. Tipicamente, um router tem um conhecimento à priori dos seus vizinhos imediatos. Um protocolo de routing partilha essa informação, de modo a que os routers tenham conhecimento da topologia da rede.

Das características específicas dos protocolos de routing fazem parte o modo como previnem a formação de ciclos ou o modo como desfazem ciclos depois de estes se formarem e também o modo como determinam o melhor caminho, a partir de uma série de possíveis métricas.

Existem três tipos básicos de protocolos de routing: “link state” e “distance vector” .

Determinação de um caminho

Uma métrica é um standard de medida, que é usado pelos protocolos de routing, para determinar o melhor caminho até ao destino. Para ajudar neste processo de determinação do caminho, os protocolos de routing inicializam e mantém tabelas, que contêm informação necessária para realizar as operações de routing. Esta informação pode variar muito, dependendo do protocolo de routing que gerou os caminhos. Quando um router recebe um pacote, verifica o seu endereço de destino e tenta associar esse endereço com o próximo salto.

Tabela de routing

Uma tabela de routing é um conjunto de regras, que definem os caminhos a seguir pelos pacotes de dados. A tabela contém a informação necessária para transmitir pacotes para o seu destino. Quando é recebido um pacote o dispositivo verifica a informação contida

nos pacotes e a informação nas tabelas de routing e define assim o caminho mais curto possível para a transmissão dos pacotes até ao seu destino.

Exemplo de uma tabela de routing

```
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
U - per-user static route
Gateway of last resort is 207.31.207.17 to network 0.0.0.0
B       212.0.212.0/24      [200/24]    via     129.250.16.133,     2d08h
B       206.102.168.0/24    [200/32]    via     129.250.16.133,     5d07h
B       206.51.253.0/24     [200/25]    via     129.250.16.133,     5d05h
B       205.204.1.0/24      [200/35]    via     129.250.16.133,     5d07h
B       204.238.34.0/24     [200/24]    via     129.250.16.133,     5d00h
B       204.17.221.0/24     [200/35]    via     129.250.16.133,     5d07h
B       199.0.199.0/24      [200/32]    via     129.250.16.133,     5d07h
B       198.17.215.0/24     [200/37]    via     129.250.16.133,     5d07h
B       194.204.14.0/24     [200/42]    via     129.250.16.133,     2d22h
B       192.153.89.0/24     [200/24]    via     129.250.16.133,     1d08h
B       192.68.132.0/24     [200/35]    via     129.250.16.133,     5d07h
B       170.170.0.0/16      [200/25]    via     129.250.16.133,     5d05h
B       208.152.73.0/24     [200/25]    via     129.250.16.133,     5d05h
B       205.152.84.0/24     [200/25]    via     129.250.16.133,     5d05h
```

O primeiro campo de cada entrada da tabela indica como foi aprendido o caminho. Alguns caminhos poderão ser adicionados através de routing estático, outros por broadcast. Os caminhos que possuem um B à frente indicam que foram aprendidos por BGP. O segundo campo de informação indica o caminho aprendido, no caso da primeira linha o caminho para os endereços de classe C 212.0.212.0/24, foi aprendido através de BGP e o próximo destino do pacote deve ser o 129.250.16.133. O último campo indica hà quanto tempo o caminho está a funcionar e estável.

Sistema Autónomo

Um sistema autónomo é um conjunto de redes e routers, cuja administração do encaminhamento é gerida pela mesma entidade. A cada AS é atribuído um identificador único de 16 bits (IANA – Internet Assigned Numbers Authority)

Protocolos de routing interno

Os protocolos de IGP (Interior Gateway Protocols) trocam informação de routing dentro de um único domínio de routing. Um sistema autónomo pode conter múltiplos domínios de routing. Como exemplo de protocolos internos temos o IGRP, EIGRP, OSPF, RIP e IS-IS.

Protocolos de routing externo

Protocolos de EGP (Exterior Gateway Protocolos) fazem o routing entre sistemas autónomos diferentes. Como exemplos de protocolos routing externo temos o EGP, BGP e CSFP.

Tipos de protocolos

Os protocolos de routing estático não podem propriamente ser chamados de protocolos, uma vez que não são mais do que tabelas criadas pelo administrador de rede, antes da rede entrar em funcionamento. Estas tabelas não se alteram, a menos que o administrador de rede as altere. Os protocolos que usam routing estático são feitos para trabalhar em ambientes onde o tráfico de rede é previsível e em redes relativamente simples.

O facto de os sistemas com routing estático não reagirem às alterações na rede, faz com que não sejam utilizados nas redes com grandes dimensões e que sofrem constantes alterações.

A maioria dos protocolos utilizados hoje em dia, são baseados em algoritmos de routing dinâmico, que se ajustam às alterações na rede, através da análise da mensagens de actualização dos routers. Se as mensagens indicarem que ocorreu uma alteração na rede, o software de routing recalcula as trajectórias e envia novas mensagens de actualização das tabelas de routing aos routers. Estas mensagens percorrem a rede indicando aos routers para executarem os seus algoritmos de modo a recalcular as tabelas de routing.

É possível complementar os algoritmos de routing dinâmicos com algoritmos de routing estáticos, sempre que se considerar necessário.

Alguns protocolos de routing mais sofisticados suportam múltiplos caminhos para o mesmo destino. Ao contrário dos algoritmos que apenas suportam um caminho, estes algoritmos de múltiplos caminhos permitem a multiplexagem de tráfego através de várias linhas. Os algoritmos de múltiplos caminhos apresentam como vantagens melhor desempenho e fiabilidade.

Métricas de routing

Os protocolos de routing possuem vários tipos de métricas, para determinar o melhor caminho. Os protocolos mais complexos podem mesmo fazer uso de várias métricas para obter o melhor resultado possível, nestes casos essas métricas são combinadas para dar origem a uma única métrica.

As métricas mais comuns são:

- Comprimento do caminho a percorrer,
- Fiabilidade,
- Atraso,

- Largura de banda,
- Carga,
- Custo da comunicação.

Comprimento do caminho a percorrer – É a métrica mais comum. Alguns protocolos de routing possibilitam aos administradores da rede, atribuir custos arbitrários a cada ligação da rede. Neste caso, o comprimento do caminho é a soma dos custos associados a cada ligação percorrida. Outros protocolos de routing definem a contagem de saltos, que é uma métrica que especifica quantos routers o pacote tem de passar, para ir da origem até ao destino.

Fiabilidade – No contexto de algoritmos de routing, refere-se ao grau de confiança de cada ligação de rede. É possível que algumas ligações de rede possam estar inoperacionais com mais regularidade que outras ou após a falha de uma rede, algumas ligações podem ser reparadas mais facilmente ou mais rapidamente que outras. Podem ser considerados vários factores para determinar o nível de fiabilidade, que não são mais que valores numéricos associados a ligações de rede, por administradores da rede.

Atraso - Refere-se à quantidade de tempo necessária para um pacote percorrer o caminho da sua origem para o seu destino, através da rede. O atraso depende de vários factores, incluindo a largura de banda, ligações de rede intermédias, as filas nas portas de cada router ao longo do caminho, congestionamento da rede em todas as ligações intermédias e a distância física a ser percorrida. Uma vez que o atraso é o resultado de um conjunto de variáveis importantes, é uma métrica de uso comum e de bastante utilidade.

Largura de banda – refere-se à capacidade de trafego disponível numa ligação. Se todos os outros parâmetros da rede forem iguais, uma ligação de 10 Mbps é preferível a uma ligação de 64 Kbps. Apesar da largura de banda ser um indicador das capacidades de uma ligação, caminhos através de ligações com maior largura de banda, não proporcionam necessariamente melhores caminhos que aqueles por ligações com menor largura de banda. Se por exemplo uma ligação rápida estiver congestionada, o tempo necessário para enviar um pacote para o destino é superior.

Carga – Refere-se ao grau de ocupação de um router. A carga pode ser calculada de várias maneiras, incluindo a utilização de CPU e pacotes processados por segundo. No entanto monitorizar estes parâmetros pode ocupar muitos dos recursos disponíveis.

Custo da comunicação – é outra métrica importante, especialmente quando algumas empresas não dão tanta importância à performance mas sim às despesas operacionais.
Apesar de o delay na linha poder ser maior, as empresas preferem enviar os pacotes pelas suas linhas em vez das linhas públicas onde se paga pelo tempo de utilização.

"Link State" e "distance Vector"

Os protocolos link state surgiram para serem utilizados em redes de grandes dimensões, o que faz com que sejam protocolos complexos, difíceis de configurar e manter.

Os algoritmos para calcular o melhor caminho não têm em conta só a largura de banda mas também outros factores.

Um factor importante em qualquer tipo de protocolo, é o tempo de convergência, que basicamente é a quantidade de tempo que a propagação de alterações de topologia demora a percorrer toda a rede.

Os protocolos link state tendem a convergir rapidamente, uma vez que apenas são comunicadas as alterações na topologia de rede aos routers vizinhos, esta informação é propagada sem alterações.

São imunes a routing loops e cada router possui a informação completa acerca da topologia.

No entanto este tipo de protocolos apresenta também algumas desvantagens: utilização de muitos recursos computacionais, elevada complexidade de cálculo da tabela de routing, a existência de vária tabelas (adjacências, topologia, routing) e a possibilidade de tráfego excessivo na descoberta inicial da topologia.

Com este tipo de protocolos, os routers trocam mensagens entre si do tipo (R, X, C), ou seja, conheço uma ligação de R para X com custo C.

Os protocolos “distance vector” foram pensados para serem utilizados em pequenas redes. São normalmente fáceis de configurar e exigem menos manutenção. Apresentam problemas quando se pretende aumentar a rede, isto porque requerem mais CPU e largura de banda. Também precisam de mais tempo para convergir.

Utilizam contagem de saltos para determinar o melhor caminho através de uma rede. A contagem de saltos é simplesmente o número de routers que o pacote tem de atravessar desde a sua origem até ao seu destino.

Escolhem sempre como melhor caminho aquele com o menor número de saltos. Isto pode ser um problema, porque o caminho com menos saltos nem sempre é o mais rápido.

As mensagens trocadas entre si pelos routers são do tipo (R, D), que significam estou a uma distância D da rede R.

Os routers trocam sempre informação sobre todos os destinos conhecidos. As mudanças propagam-se de router para router, podendo sempre existir routers com informação incorrecta. Esta troca de informação é realizada periodicamente (aproximadamente 90s). Numa rede com vários routers esta operação pode afectar o CPU e a utilização de largura de banda.

Em ambientes onde os caminhos mudem rapidamente as tabelas de routing podem não estabilizar. O algoritmo usado é lento a convergir depois de uma alteração. Todos os routers têm que entrar no processo.

Como vantagens este algoritmo possuem uma baixa complexidade do cálculo da tabela de encaminhamento, é fácil de implementar e é um protocolo largamente difundido.

Como desvantagens pode referir-se que as mensagens de update são muito extensas, as mudanças propagam-se lentamente de router para router, podendo no entanto existir routers com informação incorrecta, o algoritmo usado pode não convergir e é lento quando converge.

No fundo os algoritmos link state enviam pequenos pacotes de actualização para os seus vizinhos, enquanto os distance vector enviam grandes pacotes de actualização para toda a rede. Ao convergirem mais rapidamente, os algoritmos link state são menos propensos a loops do que os distance vector. Por outro lado, os algoritmos link state requerem menos capacidade de processamento e memória que os algoritmos distance vector, o que significa que os algoritmos link state podem tornar-se mais caros de implementar e manter. Os protocolos de routing distance vector são bastante úteis para pequenas redes, mas com redes de grandes dimensões, a melhor opção são os protocolos link-state. Os protocolos link state são difíceis de configurar e manter, mas a eficiência da rede é superior. Além disso, não sofrem dos problemas dos protocolos distance vector.

2.3 Voz sobre IP

As chamadas VoIP (Voice Over Internet Protocol) são transferências de dados digitais baseadas em pacotes, onde o meio de transferência é a internet[11]. Apesar de ser um protocolo orientado para comunicações de voz, permite também outras tecnologias como por exemplo a transmissão de imagem.
A informação presente na voz é convertida em formato digital, passando a estar disponível como pacotes de dados. Estes pacotes podem ser transmitidos pela rede de computadores, até ao computador destino, onde são reconvertidos para o formato original.

2.3.1 Vantagens e desvantagens do VoIP

Um telefone comum, utiliza a rede tradicional de comutação de circuitos, da rede pública de telefones, para transportar a voz das chamadas. O que significa que é efectuada uma ligação bidireccional entre dois telefones, durante a duração total da chamada. Durante esta ligação, vários kms de fio de cobre tem de ser mantidos para permitir que a chamada se possa realizar. Isto faz com que exista uma longa cadeia de dispositivos entre os dois telefones e também que existam recursos reservados para manter esta ligação, que não podem ser usados por outros telefones, mesmo que nenhum dos utilizadores dos telefones esteja a falar em determinado momento.

Com o VoIP, em vez de uma linha aberta, apenas é enviada informação quando realmente existe informação para enviar. Cada pacote de dados contendo informação sobre a voz é enviado por ordem, com um destinatário comum. Os pacotes passam pelos sistemas de encaminhamento e servidores. A cada pacote será indicado o caminho menos congestionado para chegar ao destino.

Isto significa que o investimento do VoIP é muito baixo, quando comparado com o da rede tradicional. O VoIP usa a Internet que é de domínio público, enquanto os telefones da rede pública, necessitam de investimentos numa série de equipamento e respectiva manutenção.

Além disso, o VoIP apresenta uma rentabilidade de ligação elevada, uma vez que quando a informação relacionada com o VoIP não está a ser transmitida, a ligação pode ser aproveitada para outro tipo de tráfego. Com o VoIP é possível aceder à net, fazer download de ficheiros, receber e-mails ao mesmo tempo que se faz uma chamada. O mesmo não acontece com a rede tradicional, onde durante uma chamada, a largura de banda continua reservada mesmo que haja silêncios por parte dos dois telefones.

Também no que diz respeito aos serviços adicionais, como colocar chamadas em lista de espera, identificação do originador da chamada, conferência, música enquanto o utilizador se encontra em espera, entre outros, costumam ser pagos quando se utiliza um telefone normal, enquanto no VoIP estes serviços são gratuitos.

As chamadas de longa distância são outros dos factores de peso do VoIP, fazer uma chamada para a China ou para o vizinho do lado, em termos de custo para o utilizador e simplesmente a mesma coisa.

Além disso, com o VoIP o utilizador encontra-se contactável em qualquer hora e em qualquer lugar, desde que possua uma ligação à Internet.

No entanto o VoIP também possui alguns problemas. A qualidade da voz não é tão boa como nos telefones tradicionais. Isto acontece porque o VoIP está sujeito aos vários problemas da Internet, como por exemplo perda de pacotes, atraso na entrega de pacotes, etc.

As falhas de energia são outro contra do VoIP, enquanto na rede tradicional, caso a energia falhe a central telefónica costuma manter-se a funcionar com geradores próprios, assegurando o funcionamento das linhas telefónicas, no caso do VoIP isto não acontece.

Também a falta de interoperabilidade entre os vários protocolos de VoIP apresenta-se como um problema.

2.3.2 Protocolos VoIP

Existem uma série de protocolos que podem ser utilizados de modo a proporcionar serviços de comunicação VoIP.

Para transmitir pacotes de vídeo e áudio entre dois computadores utiliza-se o standard Real Time Protocol (RTP). Mas antes que estes pacotes RTP possam ser transmitidos entre os dois computadores, outros protocolos têm de ser utilizados de modo a encontrar na rede o device remoto, para onde se pretendem transmitir os dados e para negociar o modo em que a informação vai circular entre os dois dispositivos. Os protocolos mais populares para este tipo de funções são o H323 e SIP (Session Initiation Protocol).

Basicamente o H323 e SIP permitem que os utilizadores façam a mesma coisa, ou seja, estabelecer uma comunicação multimedia (áudio, vídeo, etc). No entanto, o H323 e SIP diferem bastante na concepção. O H323 é um protocolo binário enquanto o SIP é um protocolo baseado em ASCII. Ambos os protocolos conseguem cumprir a função para o qual estão destinados, no entanto o H323 é superior nalguns aspectos, como por exemplo, a melhor interoperabilidade com a rede pública de telefones, melhor suporte para vídeo, transporte de DTMFs fora da banda, entre outros.

2.4 Próximos passos

Uma vez realizado este estudo, é agora possível ter alguns dados para efectuar determinadas escolhas. No próximo capítulo serão descritas as escolhas feitas ao nível do hardware e software no que diz respeito aos sub-sistemas do EPC e Encaminhador.

3

Organização da solução

3.1 Visão geral

A primeira parte desta tese pretendeu apresentar, de um modo geral, as razões que levaram o exército português a encomendar à EID uma nova versão dos Intercomunicadores de Carro de Combate (ICCs) e também os requisitos que se pretendem para essa nova versão.

Uma vez que esta tese se insere no mestrado de engenharia informática, a componente do projecto relacionada com o hardware será abordada ao de leve, focando apenas os aspectos que interessam para a componente de software, aquela que possui o maior interesse para esta pós graduação.

Esta segunda parte da tese descreve, com algum detalhe, soluções encontradas para satisfazer alguns dos requisitos pretendidos. Não será abordada toda a componente de software do projecto, serão descritos apenas os dois subsistemas, designados por Encaminhador e EPC, serão também focadas várias escolhas feitas em função das funcionalidades pretendidas para este projecto, como por exemplo, o protocolo de routing escolhido para as redes que se podem formar ao ligar vários ICCs

3.2 Opções feitas sobre o hardware

O Hardware do ICC será constituído por uma placa mãe, onde serão ligados os dois computadores integrados numa placa, responsáveis pelos dois sub-sistemas, o Encaminhador e o EPC. Do Encaminhador, para além do computador integrado numa placa, faz ainda parte uma outra placa, onde se encontram um controlador ethernet e uma ficha RJ45, correspondente à ligação ethernet do computador integrado na placa. O controlador ethernet está ligado a um switch que se encontra na placa mãe.

Na figura 3.1 é possível ver a ligação entre os dois sub-sistemas e a placa mãe.

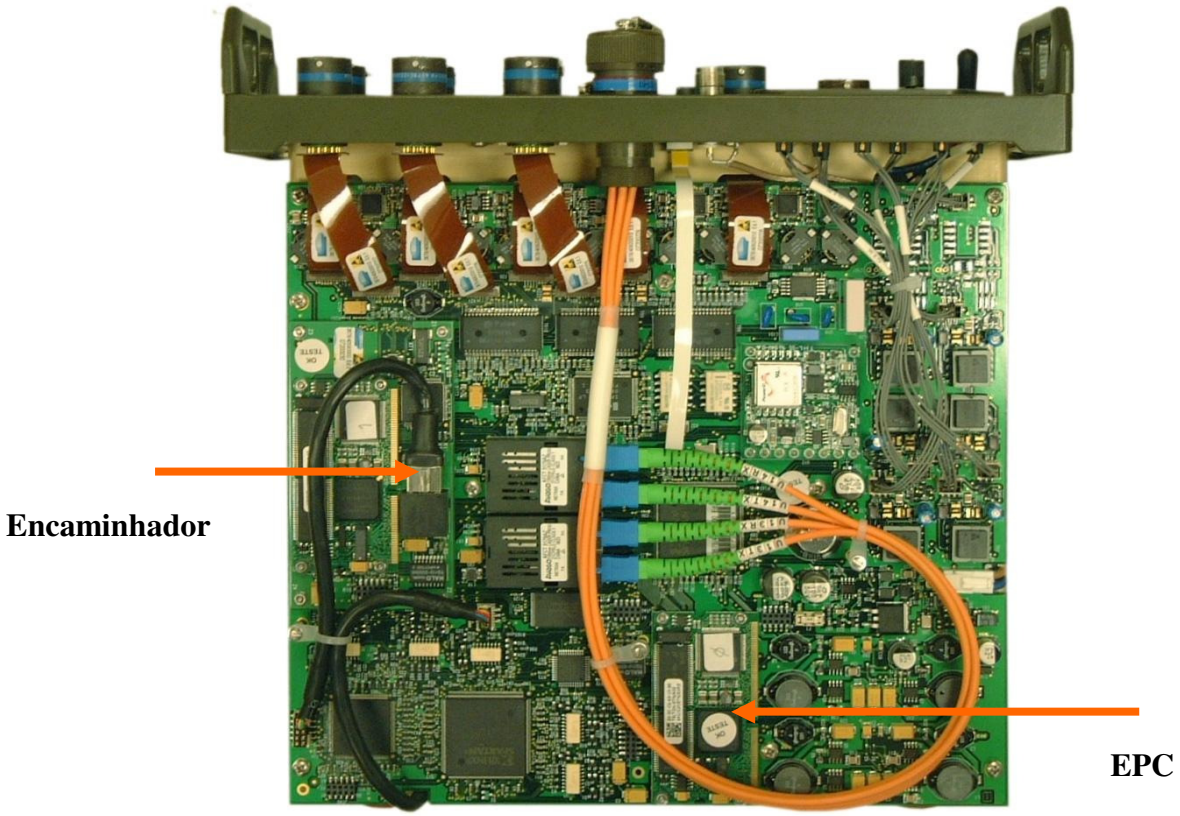


Figura 3.1 – Vista superior do ICC

A figura 3.2 mostra a ligação dos dois subsistemas dentro do ICC e também como é feita a sua ligação com os outros ICCs.

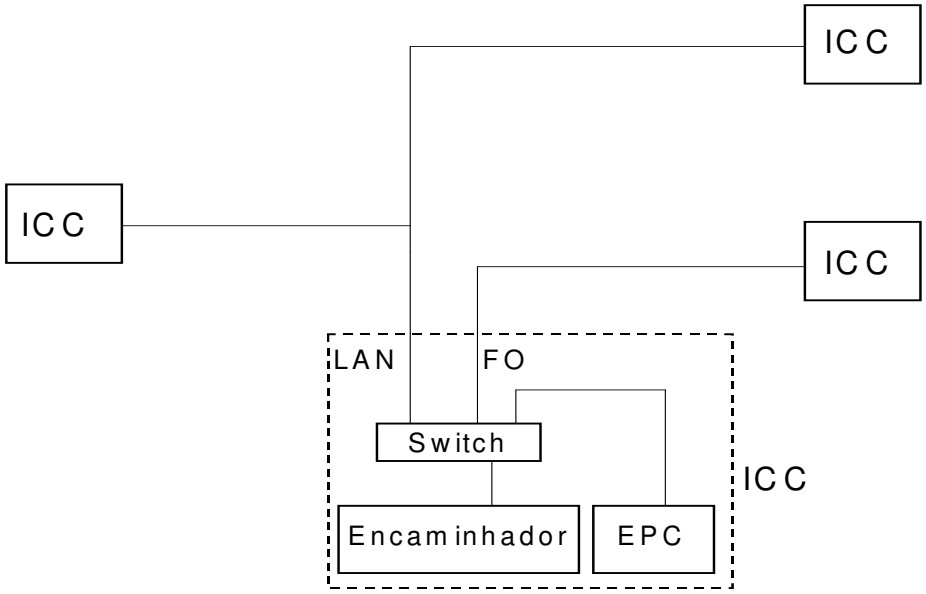


Figura 3.2 - Ligação entre os dois sub-sistemas e o exterior

3.2.1 Computador integrado numa placa

O Triton pode ser visto como um verdadeiro computador, implementado numa pequena placa, do tamanho de um cartão de crédito. Foi este pequeno computador o escolhido para operar nos dois subsistemas, Encaminhador e EPC, de modo a assegurar que ambos cumpram as funções que lhe estão destinadas.

O Triton possui um processador Xscale de 400 MHz da Intel, SDRAM e memória Flash. O Triton ocupa uma área de 67,6 * 36,6 mm com uma altura de 7,3 mm. Está optimizado para o desenvolvimento de dispositivos de Internet móvel e para aplicações de infra-estrutura de rede. O processador pode trabalhar em modo Turbo (400Mhz) ou Run (200Mhz), permitindo assim jogar com a relação do performance/consumo do processador.

As características principais do Triton são:

- Processador Intel XScaleTM PXA250 (400 MHz)
- Memória 64 MByte SDRAM (32-bit@100Mhz)
- Armazenamento de dados em 32 Mbyte Flash memory (32-bit)
- Bus I2C
- 3 interfaces série assíncronas, 1 interface serie síncrona
- Uma interface PC-CARD / compact-flash
- Uma interface JTAG
- Controlador on-board fast ethernet LAN91C111 10/100 MBit/s
- Número série único através de um DS2430A
- Consumo de 70mW (standby) / 1500mW (máximo)

Na figura 3.3 é possível ver duas fotos do Triton, uma vista de cima e uma vista de baixo.

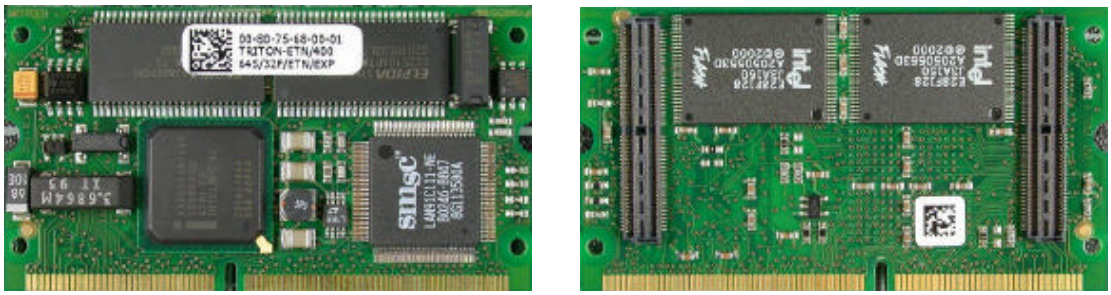


Figura 3.3 – Vista de cima e vista de baixo do Triton

A escolha do Triton foi feita tendo em conta as prestações do dispositivo, ou seja, procurando o melhor desempenho possível, mas sempre tendo em conta factores essências para este tipo de projecto, como por exemplo a manutenção do sistema. Para projectos como este, a escolha de um dispositivo sem ventoinha é considerada como obrigatório, uma vez que reduz drasticamente o número de futuras operações de manutenção, para limpeza/substituição da ventoinha, evitando assim que o ICC tenha de ser aberto periodicamente.

3.2.2 Encaminhador

Os ICCS devem possuir a capacidade de se ligarem entre eles, via ethernet, de modo a criarem uma rede onde possa circular voz (VOIP) e dados. Para permitir esta funcionalidade os ICCS devem estar munidos de um switch. Esse switch será controlado pelo processador do Encaminhador (através de uma porta MII e de um controlador ethernet) e disponibilizará ao exterior 6 portas para ligações ethernet, 4 LAN's e duas fibras óptica. Duas dessas LAN's encontram-se a funcionar em modo switching e têm como finalidade permitir que o utilizador/operador aceda ao sistema para testes, configurações, etc. As outras duas LAN's e fibras óptica, podem estar tanto no modo de switching ou routing, conforme a configuração pretendida/carregada pelo operador, existindo 16 combinações possíveis de configurações.

A figura 3.4 mostra em detalhe a ligação do Encaminhador à placa mãe.

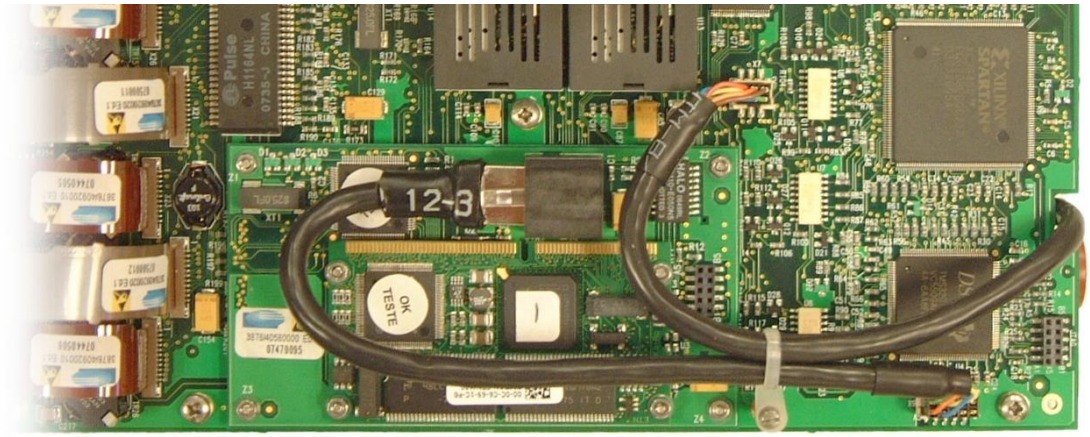


Figura 3.4 - Ligação entre o Encaminhador e placa mãe

O switch comercial apenas pode ser acedido por porta MII. Como o Triton não possui esse tipo de interface, foi necessário colocar no meio dos dois um controlador ethernet que consiga aceder ao switch via MII e que pode ser acedido pelo Triton através do bus local.

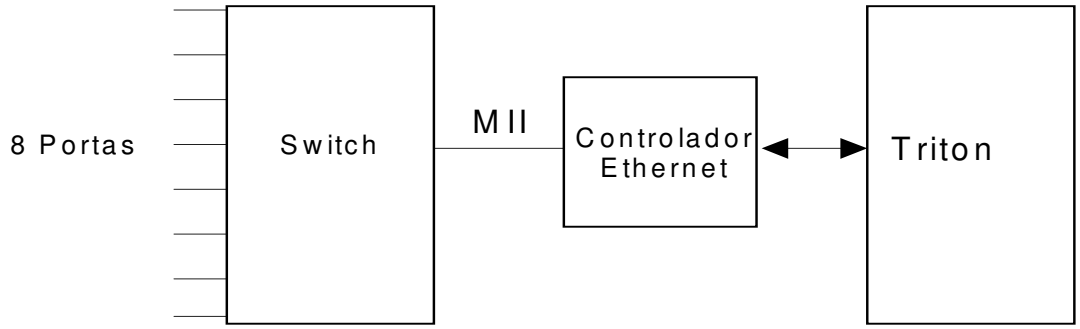


Figura 3.5 - Ligação entre Switch, Controlador e Triton

3.2.3 EPC

Enquanto o Encaminhador se encontra mais virado para as ligações de rede, o EPC deve fazer a gestão do sistema, deve saber sempre em que estado se encontra o sistema e deve saber sinalizá-lo ao operador. Deve ser o EPC a fazer a ligação entre o sistema e o software aplicacional, seja ele de manutenção, teste ou configuração.

O EPC tem também a responsabilidade de comunicar com algum hardware periférico, nomeadamente com um DSP (Digital Signal Processor), um RTC (Real Time Clock), uma FPGA, duas EEPROMs e obviamente com o Encaminhador. A comunicação com o DSP, uma das EEPROM e RTC é feita por bus I2C. A comunicação com a segunda EEPROM é feita através de um GPIO (General Purpose

Input Output), uma vez que a EEPROM possui um só fio para efectuar as comunicações com o EPC. A comunicação com a FPGA é feita através do bus local. A comunicação com o Encaminhador é feita por ethernet, através do switch do ICC.

O EPC está encarregue de carregar o software no FGPA e no DSP sempre que existe alguma actualização. A programação da FPGA é feita através do bus, enquanto a programação do DSP é colocada na EEPROM acedida através do I2C. No arranque o DSP consulta esta EEPROM para verificar se existe, ou não, uma nova versão do software.

O EPC é também o responsável pela transmissão dos pacotes VOIP, que são posteriormente encaminhados pelo router para o destino desejado.

Toda a interacção entre o ICC e os dispositivos ligados ao ICC passa pelo EPC, este deve saber como reagir aos vários inputs dos vários dispositivos e gerar os outpus respectivos, sempre que necessário.

Qualquer acesso por FTP ao EPC deve ser autenticado por username e password.

3.3 Opções feitas sobre o software

3.3.1 Sistema operativo embebido

O sistema operativo embebido escolhido, para os dois sub-sistemas, foi o Windows CE (WCE). A escolha deste sistema operativo teve como principal razão, o preço competitivo de comercialização do WCE, embora o facto de já existir alguma experiência com versões anteriores deste OS, tenha também pesado na decisão final.

O Windows CE é uma variante dos sistemas operativos Windows, mas com um kernel algo diferente das versões desktop. Feito de raiz para ser um sistema operativo pequeno e para correr em computadores minimalistas e sistemas embebidos, o Windows CE foi optimizado para dispositivos com capacidade mínima de armazenamento. Os dispositivos típicos alvo são dispositivos sem disco, onde o OS tende a ser colocado numa ROM, ou similar.

O Windows CE é um sistema operativo multi-tarefa, que obedece à definição de sistema operativo de tempo real e que suporta 256 níveis de prioridade. Um dos seus maiores atributos, para além do tamanho, é o facto de utilizar um subconjunto da API WIN32 e ser multiplataforma. O Windows CE é suportado nos processadores X86 e compatíveis, MIPS, ARM e Hitachi SuperH. Ao contrário de outros sistemas operativos Windows, o Windows CE permite o acesso ao código fonte de alguns componentes.

Os primeiros produtos a utilizar o WCE datam de 1996 e consistiam em dispositivos de organização de informação pessoal (Ex: PDAs). Hoje em dia o WCE é utilizado numa série de dispositivos tais como Pocket PCs, smartphones, reprodutores de vídeo portáteis, Smart Display, etc.

Para além de um ambiente de desenvolvimento integrado o Windows CE é também uma ferramenta de exportação de kits de desenvolvimento de software (SDKs). É possível desenvolver aplicações tanto no próprio ambiente de desenvolvimento do WCE, como no Visual Studio .Net ou no Embedded Visual C++.

O processador do Triton pertence à família dos ARM (ARMV5), que é um dos tipos de processador suportados pelo WCE.

Outro factor que pesou na escolha do Windows CE foi o facto do fabricante do Triton disponibilizar também um BSP compatível com o WCE .NET 4.2, o que facilita bastante o processo de desenvolvimento de software, uma vez que permite poupar algumas horas de trabalho a esse mesmo desenvolvimento.

O facto do BSP ser apenas compatível com o WCE .NET 4.2, fez com que fosse adquirida essa mesma versão do WCE , em detrimento do WCE 5.0, que já se encontrava à venda na altura.

3.3.2 Ferramentas de desenvolvimento do sistema operativo

Platform Builder

O Microsoft Platform Builder é um ambiente de desenvolvimento integrado (IDE) utilizado para criar plataformas, drivers e também aplicações WCE. O Platform Builder possuí todas as ferramentas necessárias para construir uma plataforma passo a passo, desde a criação ao teste. Com o Platform Builder é possível exportar SDK's (Standard Development Kit), que permitem criar aplicações, a partir de outras ferramentas que não o Platform Builder.

Embedded Visual C++ (eVC)

Esta ferramenta permite o desenvolvimento de aplicações embebidas, para dispositivos com o sistema operativo Windows CE. Para o desenvolvimento das aplicações é necessário o SDK produzido pelo Platform Builder. O Embedded VC++ é uma ferramenta que está disponível para download gratuito no site da Microsoft e que permite desenvolver aplicações em C++.

Microsoft Visual studio .NET

Esta ferramenta permite o desenvolvimento de aplicações embebidas para dispositivos com o sistema operativo Windows CE. Para o desenvolvimento das aplicações é necessário o SDK produzido pelo Platform Builder. Com o Visual Studio .Net é possível tirar vantagem da framework .NET. Sendo possível desenvolver aplicações em Visual C++, C# e Visual Basic .Net

3.3.3 Encaminhamento de pacotes entre ICCs

Uma vez que os ICCs devem conseguir ligar-se em rede, torna-se necessário escolher um protocolo de routing, que permita administrar a rede formada. O protocolo escolhido foi o RIP. Esta escolha recaiu neste protocolo essencialmente pelo número reduzido de nós das redes que os ICCs podem formar e também pela facilidade da configuração do protocolo e pelas exigências reduzidas a nível do poder de computação e capacidade de memória nos routers

O protocolo RIP (Routing Information Protocol) foi desenvolvido pela Xerox Corporation no início dos anos 80, para ser utilizado nas redes Xerox Network Systems (XNS). É um protocolo de encaminhamento baseado no algoritmo de vector-distance. Foi idealizado para ser um protocolo IGP (Interior Gateway Protocol), ou seja, um protocolo para ser usado dentro de uma rede autónoma. É um protocolo suportado por praticamente todos os fabricantes de routers.

O RIP foi desenvolvido com o objectivo de trabalhar com redes de tamanho moderado, que use uma tecnologia homogénea. Não foi criado para ser utilizado em ambientes complexos. O protocolo RIP funciona bem em pequenos ambientes, mas apresenta sérias limitações quando utilizado em grandes redes. O número de saltos (Hops) entre hosts é limitado, apenas 15 (16 é considerado infinito). O protocolo é limitado a redes em que o caminho mais longo use 15 saltos. Qualquer destino com custo superior a 15 saltos é indicado como tendo um custo de infinito. No caso de uma rede grande, caso se forme um loop, a resolução do loop pode demorar bastante tempo. Este protocolo usa métricas fixas para comparar rotas alternativas. Não é apropriado para situações onde as rotas tenham de ser escolhidas com base em parâmetros de tempo real, como delay medido, fiabilidade ou tráfego. Os elementos dentro de uma rede RIP devem enviar mensagens de actualização da rede, com informação das redes e sub-redes que alcança, de 30 em 30 segundos. Isto faz do protocolo um consumidor da largura de banda. Se um dos elementos passar 180 segundos sem receber mensagens de actualização de outro elemento conhecido, assume que o outro elemento não está operacional ou a ligação entre eles não funciona.

3.3.4 Suporte de voz sobre IP

No que diz respeito ao VOIP optou-se por deixar de lado os típicos protocolos de negociação (SIP e H323). Assim que o ICC detecta uma configuração, que indique que uma das suas portas está em routing, inicia um processo de envio de mensagens por essa porta, na expectativa de encontrar um outro ICC. Após detectar outro ICC, é feita uma negociação de quais os IPs a utilizar (caso não tenham sido definidos como sendo do tipo estático), uma vez decididos e adicionados os IPs aos ICCs, os ICCs passam a estar ligados em rede. Assim que esta ligação se encontrar activa, o Encaminhador notifica o EPC da existência de um outro ICC ligado às suas portas e o EPC inicia então o envio dos pacotes RTP.

3.3.5 Device Drivers

Foram desenvolvidos device drivers para os dois componentes do sistema:

Device driver do Encaminhador

Um dos requisitos exigidos para os ICCs é que se consigam ligar em rede entre eles, para tal dispõem de 4 portas Lan e 2 portas fibra óptica, estas 6 portas são controladas por um switch, que é controlado e acedido pelo router por meio de um controlador ethernet. Para tal, o OS do router necessita de um driver que permita interagir com o switch (através do controlador), de modo a que se consiga estabelecer uma ligação entre o StackIP do WCE e o switch. O WCE possui um driver específico para estas situações, chama-se NDIS driver.

Device Drivers do EPC

Foram desenvolvidos 3 device drivers para o sistema operativo do EPC:

- um device driver para comunicar com o DS2430A
- um device driver para comunicar com o DSP (Digital Signal Processor), RTC (Real Time Clock) e EEPROM (Electrically-Erasable Programmable Read-Only Memory) através de I2C.
- um device driver para comunicar com a FPGA

Realização da solução

No capítulo seguinte descreve-se detalhadamente a realização dos device drivers bem como de outro software desenvolvido.

4

Implementação da solução

Neste capítulo começa-se por descrever em detalhe o hardware utilizado. Seguidamente faz-se uma descrição do software desenvolvido, começando por discutir os device drivers do Windows CE (WCE).

4.1 Hardware utilizado

Tal como havia sido referido no capítulo anterior, o subsistema EPC é constituído por um computador integrado numa placa, enquanto o subsistema Encaminhador é constituído por um computador integrado numa placa e uma outra placa, onde o computador é ligado, que contém um controlador ethernet e uma ficha RJ45 que está associada à ethernet do computador.

A escolha do Triton como peça central do hardware já foi devidamente justificada no capítulo anterior, bem como descritas as suas características.

Quanto ao controlador ethernet, foi necessário recorrer à sua utilização, porque o switch escolhido apenas permite o acesso por porta MII. Como o Triton não possui este tipo de porta, foi necessário recorrer a um Ethernet Controller para servir de

intermediário entre o Triton e o switch. É através da porta MII que se faz a configuração do switch e através da qual se tem acesso aos dados recebidos da rede.

4.1.1 I2C

O bus I2C foi criado pela Philips Corporation e consiste num bus série com uma interface de 2 pinos e que serve para trocar informação entre dispositivos. O pino de dados, designado por SDA é utilizado para input e output de dados e o pino de clock SCL, é utilizado para controlo e referência do bus I2C. O processador tanto pode funcionar como master ou slave, o que significa que qualquer um dos pinos é bidireccional. A unidade I2C é um dispositivo periférico, que reside no bus interno do processador. Os dados são transmitidos e recebidos pelo bus I2C através de uma interface com um buffer. O controlo e status do bus é consultado através de uma série de registos mapeados em memória. Cada dispositivo é identificado no bus através de um endereço de 7 bits e qualquer dispositivo pode receber ou enviar informação e actuar como master ou slave.

O bus I2C permite o funcionamento com múltiplos masters, o que significa que mais que um device pode iniciar a transmissão de dados ao mesmo tempo. Para suportar esta característica, o bus I2C possui um esquema de arbitragem.

O mecanismo de interrupção do processador pode ser usado para notificar o CPU que existe actividade no bus I2C. Em vez de interrupts pode utilizar-se a técnica de polling. O bus I2C consiste num buffer de 8 bits para passar dados para e do processador, uma série de registos de controlo e de status e um shift register para conversões série/paralelo. A unidade I2C sinaliza interrupts quando o buffer está cheio, quando o buffer está vazio, quando o endereço de slave da unidade é detectado, quando existe perda de arbitragem ou quando existe uma condição de erro no bus. O I2C suporta o funcionamento em dois modos, o rápido (400 Kbits/sec) e o modo standard (100 Kbits/sec).

Para comunicar com um slave, o master tem enviar um primeiro byte que consiste no endereço de slave do dispositivo, que o master pretende contactar. O endereço dos dispositivos é de 7 bits, o 8º bit do buffer também é preenchido e indica se o master pretende escrever ou ler. O Slave responde com um acknowledge. A partir daqui o master indica o endereço onde pretende escrever ou ler e procede à escrita ou leitura, conforme o pretendido.

Toda a informação aqui descrita sobre o I2C, incluindo as várias imagens dos registos e sinalização, foi retirada do manual do processador do Triton[14].

Estados de Start e Stop e não Start/não Stop

A especificação define uma transição de Start, utilizada sempre que se inicia uma transferência, e uma transição de Stop, utilizada sempre que se acaba uma transferência. Uma condição de Start dá-se quando acontece uma transição de high para low na linha SDA quando SCL está high. Uma condição de Stop dá-se quando acontece uma transição de low para high na linha SDA quando SCL está high.

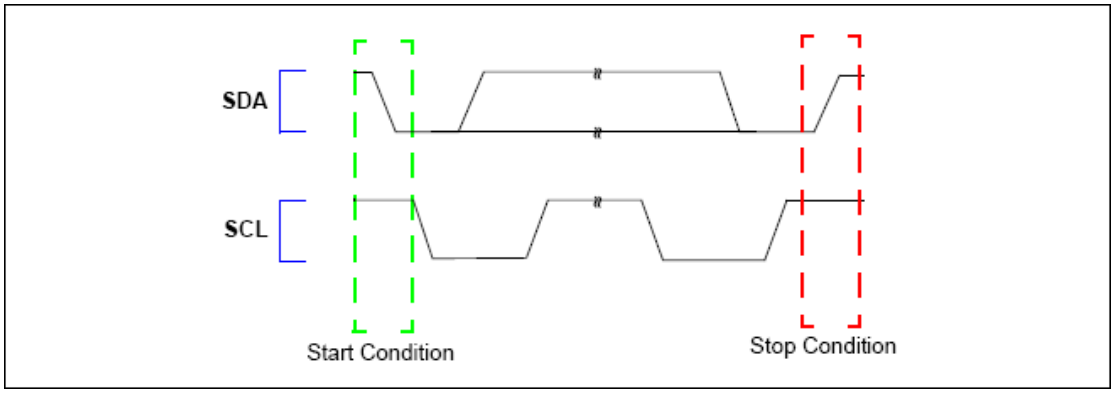


Figura 4.1 – Stop condition e Start condition

Condição de Start

A condição de Start inicia uma transação do master ou um start repetido. O software deve carregar o endereço do target e o bit que indica se o master pretende fazer uma leitura ou escrita. Um start repetido permite o master fazer múltiplas transferências para diferentes slaves sem libertar o bus.

Condição de não-start e não-stop

Esta condição é utilizada pelo master em modo de transmissão, quando o I2C está a transmitir múltiplos bytes de dados.

Condição de Stop

A condição de Stop termina uma transferência.

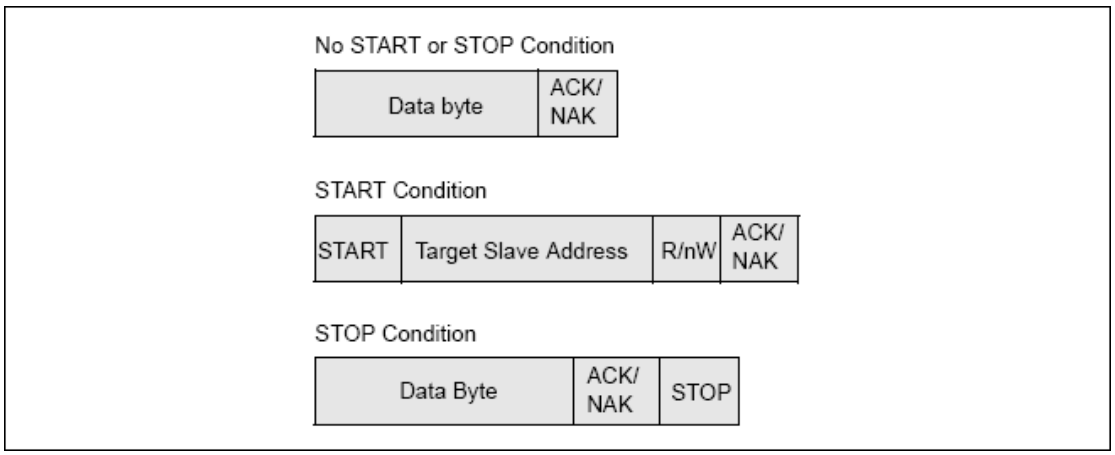


Figura 4.2 – Start, Stop e no Start or Stop condition

Endereçamento de um Slave

Como Master, a unidade I2C deve compor e enviar o primeiro byte da transacção. Este byte consiste no endereço do slave que pretende contactar e o bit que indica operação escrita ou leitura. O bit mais significativo é transmitido primeiro.

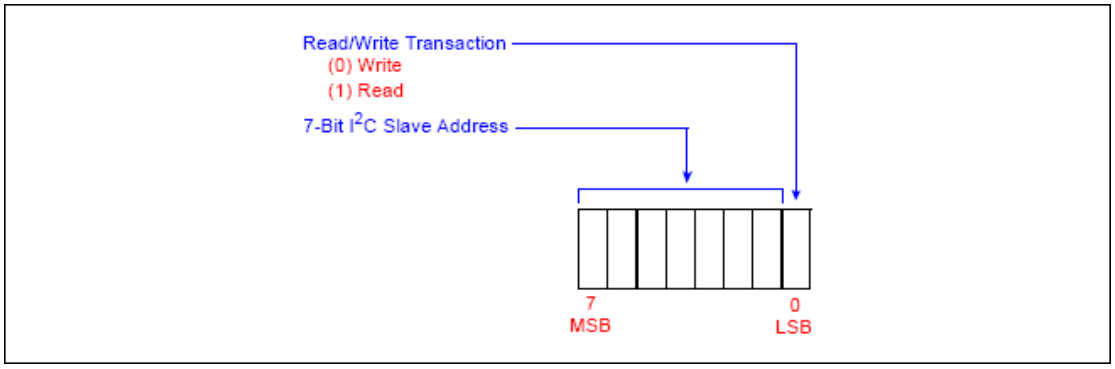


Figura 4.3 – Endereço do Slave e operação pretendida

O primeiro byte a ser transmitido é seguido de um ACK do slave endereçado. Quando é retornado um NAK a unidade I2C aborta a transacção enviando automaticamente um STOP.

Acknowledge no I2C

Qualquer transferência de bytes é acompanhada de um ACK que o master ou o slave geram. Quem está em modo de transmissão deve libertar a linha SDA para que quem está em modo de receptor possa enviar o acknowledge.

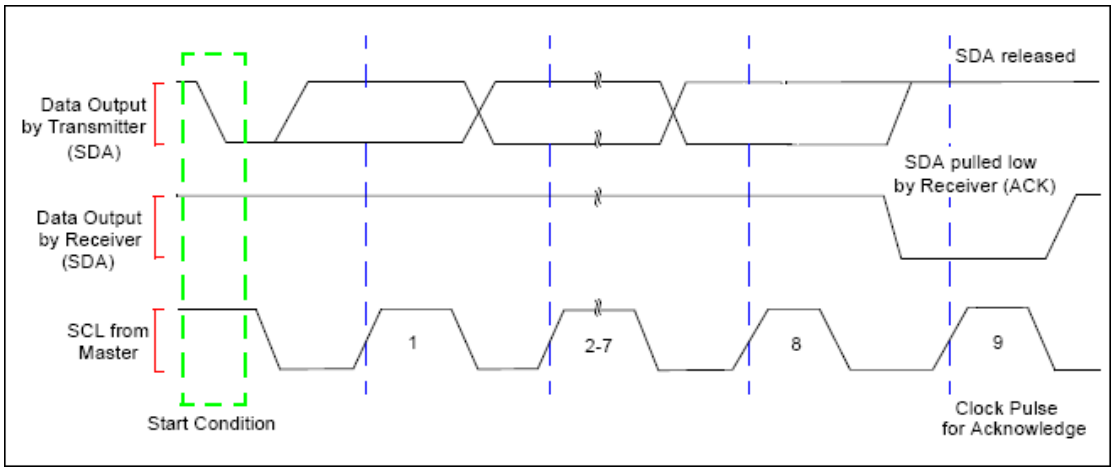


Figura 4.4 – Acknowledge no I2C

Arbitragem

O bus I2C possuí capacidades multimaster, o que requer um modo de arbitragem. A arbitragem aparece quando dois ou mais masters tentam iniciar uma comunicação.

Numa situação de colisão, se os bits de endereços, que indica se o master pretende uma escrita ou leitura, de dois masters forem iguais, a arbitragem espera pelos bits de dados para decidir quem perde a arbitragem. O master que perde arbitragem retira os sinais de SDA e SDL dos seus drivers de dados e passa novamente ao mode de recepção-slave.

Definição dos registos

IBMR

O registo IBMR indica o estado das linhas SCL e SDA.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																														SCLS	SDAS
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

31:2	—	reserved
1	SCLS	SCL Status: This bit continuously reflects the value of the SCL pin.
0	SDAS	SDA Status: This bit continuously reflects the value of the SDA pin.

Figura 4.5 – Registo IBMR

IDBR

O processador utiliza o registo IDBR para transmitir e receber dados do I2C. O registo recebe dados vindos da unidade I2C depois de um byte ser recebido e ter sido dado o acknowledge.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																								IDB							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31:8	—	reserved
7:0	IDB	**I²C Data Buffer**: Buffer for I²C bus send/receive data.

Figura 4.6 – Registo IDBR

ICR

O processador utiliza este registo para controlar a unidade I2C.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																FM	UR	SADIE	ALDIE	SSDIE	BEIE	IRFIE	ITEIE	GCD	IUE	SCLE	MA	TB	ACKNAK	STOP	START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

31:16	—	reserved
15	FM	**Fast Mode:** 0 = 100 KBit/sec. operation 1 = 400 KBit/sec. operation
14	UR	**Unit Reset**: 0 = No reset. 1 = Reset the I^2C unit only.
13	SADIE	**Slave Address Detected Interrupt Enable**: 0 = Disable interrupt. 1 = Enables the I^2C unit to interrupt the processor when it detects a slave address match or general call address.
12	ALDIE	**Arbitration Loss Detected Interrupt Enable**: 0 = Disable interrupt. 1 = Enables the I^2C unit to interrupt the processor when it loses arbitration in master mode.
11	SSDIE	**Slave STOP Detected Interrupt Enable**: 0 = Disable interrupt. 1 = Enables the I^2C unit to interrupt the processor when it detects a STOP condition in slave mode.

Figura 4.7 – Registo ICR, bits 11 a 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																FM	UR	SADIE	ALDIE	SSDIE	BEIE	IRFIE	ITEIE	GCD	IUE	SCLE	MA	TB	ACKNAK	STOP	START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

10	BEIE	**Bus Error Interrupt Enable**: 0 = Disable interrupt. 1 = Enables the I²C unit to interrupt the processor for the following I²C bus errors: • As a master transmitter, no ACK was detected after a byte was sent. • As a slave receiver, the I²C unit generated a NAK pulse. **NOTE:** Software is responsible for guaranteeing that misplaced START and STOP conditions do not occur. See Section 9.7.
9	IRFIE	**IDBR Receive Full Interrupt Enable**: 0 = Disable interrupt. 1 = Enables the I²C unit to interrupt the processor when the IDBR receives a data byte from the I²C bus.
8	ITEIE	**IDBR Transmit Empty Interrupt Enable**: 0 = Disable interrupt. 1 = Enables the I²C unit to interrupt the processor after transmitting a byte onto the I²C bus.
7	GCD	**General Call Disable**: 0 = Enables the I²C unit to respond to general call messages. 1 = Disables I²C unit response to general call messages as a slave. Must be set when the I²C unit sends a master mode general call message.
6	IUE	**I²C Unit Enable**: 0 = Disables the unit and does not master any transactions or respond to any slave transactions. 1 = Enables the I²C unit (defaults to slave-receive mode). Software must ensure that the I²C bus is idle before it sets this bit.
5	SCLE	**SCL Enable**: 0 = Disables the I²C unit from driving the SCL line. 1 = Enables the I²C clock output for master mode operation.
4	MA	**Master Abort**: generates a STOP without transmitting another data byte when the I²C unit is in master mode. 0 = The I²C unit transmits STOP using the STOP ICR bit only. 1 = The I²C unit sends STOP without data transmission. In master-transmit mode, after a data byte is sent, the ICR's Transfer Byte bit is cleared and IDBR Transmit Empty bit is set. When no more data bytes need to be sent, setting master abort bit sends the STOP. The Transfer Byte bit (03) must remain clear. In master-receive mode, when a NAK is sent without a STOP (STOP ICR bit was not set) and the processor does not send a repeated START, setting this bit sends the STOP. Once again, the Transfer Byte bit (03) must remain clear.
3	TB	**Transfer Byte**: used to send/receive a byte on the I²C bus. 0 = Cleared by I²C unit when the byte is sent/received. 1 = Send/receive a byte. The processor can monitor this bit to determine when the byte transfer is completed. In master or slave mode, after each byte transfer, including ACK/NAK bit, the I²C unit holds the SCL line low (inserting wait states) until the Transfer Byte bit is set.

Figura 4.8 - Registo ICR bits 3 a 10

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																FM	UR	SADIE	ALDIE	SSDIE	BEIE	IRFIE	ITEIE	GCD	IUE	SCLE	MA	TB	ACKNAK	STOP	START
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

2	ACKNAK	**ACK/NAK Control**: defines the type of ACK pulse sent by the I²C unit when in master-receive mode. 0 = The I²C unit sends an ACK pulse after it receives a data byte. 1 = The I²C unit sends a negative ACK (NAK) after it receives a data byte. The I²C unit automatically sends an ACK pulse when it responds to its slave address or when it responds in slave-receive mode, independent of the ACK/NAK control bit setting.
1	STOP	**STOP**: initiates a STOP condition after the next data byte on the I²C bus is transferred in master mode. In master-receive mode, the ACK/NAK control bit must be set along with this bit. See Section 9.3.3.3 for more details on the STOP state. 0 = Do not send a STOP. 1 = Send a STOP.
0	START	**START**: initiates a START condition to the I²C unit when in master mode. See Section 9.3.3.1 for more details on the START state. 0 = Do not send a START. 1 = Send a START.

Figura 4.9 - Registo ICR, bits 0 a 2

ISR

O ISR sinaliza os interrupts para o controlador de interrupts do processador. O software pode utilizar o ISR para verificar o estado da unidade I2C e bus. O registo é actualizado após o ACK/NAK ter sido recebido/enviado.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																					BED	SAD	GCAD	IRF	ITE	ALD	SSD	IBB	UB	ACKNAK	RWM
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31:11	—	reserved
10	BED	**Bus Error Detected**: 0 = No error detected. 1 = The I²C unit sets this bit when it detects one of the following error conditions: • As a master transmitter, no ACK is detected on the interface after a byte is sent. • As a slave receiver, the I²C unit generates a NAK pulse. **NOTE:** When an error occurs, I²C bus transactions continue. Software must ensure that misplaced START and STOP conditions do not occur. See Section 9.4.4. To clear this bit, write a 1 to it.
9	SAD	**Slave Address Detected**: 0 = No slave address detected. 1 = I²C unit detected a 7-bit address that matches the general call address or ISAR. An interrupt is signalled when the SADIE interrupt is set to a 1. To clear this bit, write a 1 to it.
8	GCAD	**General Call Address Detected**: 0 = No general call address received. 1 = I²C unit received a general call address.
7	IRF	**IDBR Receive Full**: 0 = The IDBR has not received a new data byte or the I²C unit is idle. 1 = The IDBR register received a new data byte from the I²C bus. An interrupt is signalled when the IRFIE is set to a 1. To clear this bit, write a 1 to it.
6	ITE	**IDBR Transmit Empty**: 0 = The data byte is still being transmitted. 1 = The I²C unit has finished transmitting a data byte on the I²C bus. An interrupt is signalled when the ITEIE interrupt is set to 1. To clear this bit, write a 1 to it.
5	ALD	**Arbitration Loss Detected**: used during multi-master operation. 0 = Cleared when arbitration is won or never took place. 1 = Set when the I²C unit loses arbitration. To clear this bit, write a 1 to it.
4	SSD	**Slave STOP Detected**: 0 = No STOP detected. 1 = Set when the I²C unit detects a STOP while in slave-receive or slave-transmit mode. To clear this bit, write a 1 to it.
3	IBB	**I²C Bus Busy**: 0 = I²C bus is idle or the I²C unit is using the bus (i.e., unit busy). 1 = Set when the I²C bus is busy but the I²C unit is not involved in the transaction.
2	UB	**Unit Busy**: 0 = I²C unit not busy. 1 = Set when the I²C unit is busy. Defined as the time between the first START and STOP.

Figura 4.10 – Registo ISR, bits 2 a 31

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																					BED	SAD	GCAD	IRF	ITE	ALD	SSD	IBB	UB	ACKNAK	RWM
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

1	ACKNAK	**ACK/NAK Status**: 0 = I²C unit received or sent an ACK on the bus. 1 = I²C unit received or sent a NAK. Used in slave-transmit mode to determine when the transferred byte is the last one. Updated after each byte and ACK/NAK information is received.
0	RWM	**Read/Write Mode**: 0 = I²C unit is in master-transmit or slave-receive mode. 1 = I²C unit is in master-receive or slave-transmit mode. R/nW bit of the slave address. Automatically cleared by hardware after a stop state.

Figura 4.11 – Registo ISR bits 0 e 1

ISAR

Este registo define o endereço slave do dispositivo. Quando em modo slave-receive, o processador responde quando os 7 bits do endereço slave do byte recebido forem iguais aos deste registo. Este registo é preenchido antes de colocar o I2C em modo activo.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
reserved																									ISA						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

31:7	—	reserved
6:0	ISA	**I²C Slave Address**: 7-bit address that the I²C unit responds to when in slave-receive mode.

Figura 4.12 – Registo ISAR

4.1.3 Boot loader do Triton

Para carregar a imagem do WCE no Triton foi necessário carregar primeiro o boot loader compatível. Esse boot loader é carregado através da porta JTAG e foi fornecido com o BSP adquirido. Só depois de carregado o boot loader, é então possível carregar a imagem do WCE via ethernet. O código fonte do boot loader é fornecido pelo fabricante do Triton, o que permite que seja feitas alterações sempre que necessário.

Durante o arranque do dispositivo, depois de carregado o boot loader via JTAG este encarrega-se de enviar, via porta série, um menu que permite a configuração de alguns parâmetros, como por exemplo o endereço IP e MAC da placa de rede on-board ou o modo do processador (Run/Turbo).

4.1.4 Controlador de ethernet

Para aceder aos dados do switch e configurá-lo, foi necessário encontrar um controlador de ethernet. A escolha recaiu sobre o LAN9115, que para além de ter um preço competitivo obedecia aos requisitos pretendidos.

O LAN9115 é um controlador ethernet 10/100 Mbps integrado num único chip, orientado para sistemas embebidos. O LAN9115 foi concebido a pensar em aplicações com requisitos médios de performance. O bus permite a ligação a microprocessadores e microcontroladores de 16 bits, assim como microprocessadores de 32 bits que conseguem expor um bus externo de 16 bits.

O LAN9115 possui FIFOs de transmissão e recepção de tamanho considerável, que permitem trabalhar com aplicações com grandes latências.

O LAN9115 possui características que reduzem ou eliminam a perda de pacotes. A sua SRAM interna permite guardar 200 pacotes recebidos. Se o fifo de recepção encher, o LAN9115 pode gerar automaticamente pacotes de controlo de fluxo, direccionados ao originador dos pacotes.

4.1.5 Switch escolhido para o ICC

O switch escolhido para os ICCs foi o IP178C. Este switch está colocado na placa mãe e é acedido pelo controlador de ethernet através de uma porta MII. Possui 9 portas, SSRAM e 8 ethernet transceveirs de 10/100 Mbps. Cada transceiver possui elevada imunidade ao ruído. Pode operar em modo store and forward. Cada porta pode ser configurada para auto-negociação, ou forçada a 10 ou 100 Mbps. Suporta ainda duas portas de fibra óptica.

4.2 Software Desenvolvido

O software desenvolvido para ambos os subsistemas resume-se a uma série de devices drivers. Um device driver para o Encaminhador e cinco para o EPC. Todos os device drivers foram desenvolvidos e testados através da ferramenta Platform Builder.

Antes de serem apresentados os detalhes de implementação será feita uma introdução aos device drivers em WCE, com especial atenção aos tipos de device drivers utilizados nos dois sub-sistemas.

De notar que a maior parte dos device drivers desenvolvidos comunica com os periféricos através do bus I2C, tendo já sido feita uma breve apresentação das características e funcionamento deste tipo de bus.

4.3 Device Drivers para WCE

Os Device Drivers são módulos que proporcionam a interface entre o sistema operativo e o hardware. No Windows CE os device drivers estão divididos em cinco grandes grupos[15]:

- native
- bus
- stream interface

- USB
- NDIS

Os device drivers native, por vezes chamados built-in drivers, são drivers exigidos pelo hardware e que foram criados pelo OEM (Original Equipment Manufacturer) quando o hardware do Windows CE foi projectado. Como exemplos de devices native temos os devices do teclado, touch panel e áudio. Estes devices drivers podem não se enquadrar na interface genérica de um device driver WCE, podendo as diferenças consistir apenas numa extensão da interface tradicional ou então ser radicalmente diferente desta.

Os device drivers bus controlam os buses do sistema, tais como o bus PCI, PCMCIA, compact flash ou dos slots SDIO (Secure Digital Input Output). Os drivers bus fazem a descoberta do hardware existente no bus, e determinam os recursos (endereços, níveis de interrupção) que estão ocupados e os que é necessário reservar. Os device drivers bus indicam também ao device manager quais os drivers apropriados a carregar, para o hardware presente no bus.

Os device drivers Stream interface dão suporte a suporte a hardware extra que exista no sistema como portas série e paralelo.

O bus série universal (Universal Serial Bus - USB) é um protocolo de comunicação que suporta transferências de dados em série de alta velocidade, entre um único host e vários periféricos USB, como sejam teclado e rato.

Os drivers NDIS permitem o acesso ao hardware de rede (ethernet, IrDA) ao código de sistema que suporta os protocolos de nível "data link" e rede.

4.3.1 Nomes dos drivers

Os stream interface drivers são identificados com um nome de três caracteres seguidos de um dígito e de (:), como por exemplo "COM2:", permitindo este esquema de designação instalar no WCE um máximo de 10 device drivers com o mesmo nome, uma vez que o número das instâncias varia de 0 a 9. Para referenciar um driver interface, as aplicações usam os 3 caracteres do nome, seguido do digito e seguido de (:). O (:) é obrigatório no WCE de modo a que o sistema reconheça o device driver.

Os drivers bus não costumam ter um nome com três letras. Como consequência os drivers bus não estão acessíveis a aplicações, como estão os stream drivers. No entanto os bus drivers são carregados pelo device manager e na maior parte dos casos são carregados e tratados como stream drivers.

4.3.2 Registry

O registry possui uma lista dos drivers a carregar e cada um destes drivers está relacionado com uma chave que possui quatro valores associados. Estes quatro valores são as quatro entradas básicas utilizadas por um device driver no Windows CE e são eles DLL, Order, Prefix e Index. O campo DLL especifica o nome da DLL

que implementa o driver. O campo Order varia entre 0 e 255 e especifica a ordem pela qual o device é carregado. O registry carrega primeiro os devices com valores de Order mais baixos. O valor do Prefix define as 3 letras que identificam o driver. Este valor é obrigatório para stream drivers, mas não para bus drivers. O valor Index indica qual a instância do driver. Quando um aplicação pretende aceder ao driver deve utilizar o Prefix juntamente com o Index.

4.3.3 Entry points

Um driver Stream Interface pode ter até 10 entry points: Init, DeInit, Open, Close, Read, Write, Seek, PowerUp, PowerDown e IOControl. Destes 10 entry points Init, DeInit, Open, Close são obrigatórios juntamente com um dos Read, Write, Seek ou IOControl; se os entry points obrigatórios não existirem o carregamento do driver falha. Na descrição seguinte, a cadeia de caracteres xxx que antecede as funções deve ser substituída pelos três caracteres que identificam o driver (Ex: COM, LPT, etc).

xxx_Init - Chamado quando uma instância do driver é carregada. O hardware a ser usado deve ser testado e inicializado. O resultado de cada Init deve ser independente para cada instância. A função é declarada do seguinte modo:

```
DWORD xxx_Init (DWORD dwContext);
```

DwContext identifica a instância. Se a função devolver zero isso indica que houve um erro.

xxx_Deinit - Chamado quando uma instância do driver é descarregada. O hardware que estava a ser utilizado deve ser desligado ou colocado num estado passivo. Qualquer memória que tenha sido reservada durante o Init deve ser libertada. A função é declarada do seguinte modo:

```
BOOL xxx_Deinit (DWORD dwContext);
```

DwContext identifica a instância.

xxx_Open - Chamado no driver quando uma aplicação chama CreateFile() sobre o driver em questão. A função é declarada do seguinte modo:

```
DWORD xxx_Open (DWORD dwContext, DWORD AccessCode, DWORD
ShareMode);
```

DwContext identifica a instância. O parâmetro AccessCode indica se o driver é aberto para leitura ou leitura e escrita. ShareMode indica se o driver é aberto com possibilidade de partilhar a abertura na escrita ou leitura, ou ambos. Se a função devolver zero isso indica que houve um erro.

xxx_Close - Chamado no driver quando uma aplicação chama CloseHandle() sobre o driver em questão. Qualquer memória que tenha sido reservada durante o Open deve ser libertada. A função é declarada do seguinte modo:

BOOL xxx_Close (DWORD dwContext);

DwContext identifica a instância.

xxx_Read - Chamado no driver quando uma aplicação chama ReadFile() sobre o driver em questão. A função é declarada do seguinte modo:

DWORD xxx_Read (DWORD dwContext, LPVOID pBuffer, DWORD Count);

DwContext identifica a instância. pBuffer é um apontador para o buffer da aplicação, onde devem ser colocados os dados. Count indica o número de bytes a ler para o buffer. A função deve retornar o número de bytes lidos. Se a função devolver zero isso indica que houve um erro.

xxx_Write - Chamado no driver quando uma aplicação chama WriteFile() sobre o driver em questão. A função é declarada do seguinte modo:

DWORD xxx_Write (DWORD dwContext, LPCVOID pBuffer, DWORD Count);

DwContext identifica a instância. pBuffer é um apontador para o buffer da aplicação, de onde devem ser retirados os dados para escrita. Count indica o número de bytes a escrever. A função deve retornar o número de bytes escritos. Se a função devolver zero isso indica que houve um erro.

xxx_Seek - Chamado no driver quando uma aplicação chama SetFilePointer() sobre o driver em questão. A função é declarada do seguinte modo:

DWORD xxx_Seek (DWORD dwContext, long Amount, WORD Type);

DwContext identifica a instância. O campo Type identifica o tipo de seek. Existem três tipos de Seek: FILE_BEGIN - pesquisa desde o início, FILE_CURRENT - pesquisa desde a posição corrente e FILE_END – pesquisa desde o final. O campo Amount indica quanto deve avançar o apontador a partir do tipo de seek pretendido.

xxx_PowerDown – Chamado antes do sistema ser suspenso. A função é declarada do seguinte modo:

VOID xxx_PowerDown (DWORD dwContext);

DwContext identifica a instância. O hardware deve ser desligado ou colocado num modo passivo.

xxx_PowerUp – Chamado antes do sistema passar a resume. A função é declarada do seguinte modo:

VOID xxx_PowerUp (DWORD dwContext);

DwContext identifica a instância. O hardware pode ser reinicializado ou colocado em modo activo..

xxx_IOControl - Chamado no driver quando uma aplicação chama DeviceIoControl() sobre o driver em questão. A função é declarada do seguinte modo:

VOID xxx_IoControl (DWORD dwContext. DWORD dWCode, PBYTE pBufIn, DWORD dwLenIn, PBYTE pBufOut, DWORD dwLenOut, PDWORD pdwActualOut);

DwContext identifica a instância. DwCode identifica a razão da chamada ao IoControl. PBufIn e dwLenIn descrevem o buffer passado ao IoControl, o primeiro é o apontador para o buffer o segundo o tamanho do buffer. PBufOut e dwLenOut descrevem o buffer que o IoControl pode preencher, o primeiro é o apontador para o buffer o segundo o tamanho do buffer. Os buffer e respectivos tamanhos não são parâmetros obrigatórios. PdwActualOut indica o número de bytes efectivamente colocados no buffer de output, caso tenha sido colocado algum.

4.3.4 NDIS Driver

Um dos requisitos exigidos para os ICCs é que se consigam ligar em rede entre eles, e para tal estão disponíveis 4 portas Lan e 2 portas fibra óptica; estas 6 portas são controladas por um switch, que é controlado e acedido pelo router. Para tal o OS do router necessita de um driver que permita aceder ao switch, de modo a que se consiga estabelecer uma ligação entre o Stack IP do WCE e o switch. O WCE possui um driver específico para estas situações chamado NDIS driver.

Os drivers NDIS podem ser divididos em três tipos básicos:

Miniport drivers - Um miniport driver trabalha directamente com a placa de rede (NIC – Network Interface Card) e proporciona uma interface a drivers de mais alto nível.

Intermediate drivers - Um intermediate driver trabalha entre os drivers de mais alto nível, protocol drivers, e os de mais baixo nível miniport drivers.

Protocol drivers – Um protocol driver implementa na sua parte alta, uma Transport Driver Interface (TDI) ou outra interface específica de uma aplicação que proporciona serviços aos utilizadores da rede. Por outro lado, a sua parte baixa, disponibiliza uma interface de protocolo com driver de nível mais baixo, que permite receber e enviar pacotes. O sistema operativo fornece protocol drivers para TCP/IP, Point-to-Point Protocol (PPP) e Infrared Data Association (IrDA).

A figura 4.13 apresenta a arquitectura do protocol driver[18] e a 4.14 a arquitectura do driver NDIS[19].

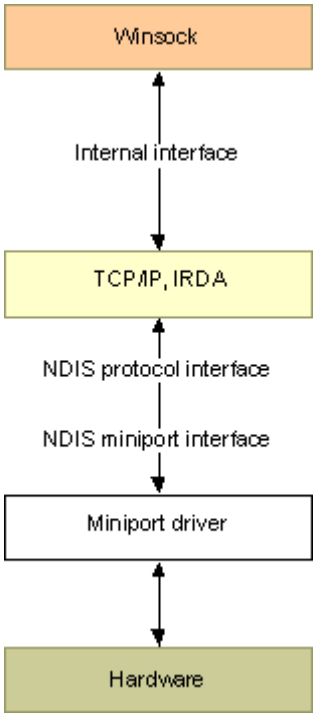


Figura 4.13 - Arquitectura do protocol driver

Destes três tipos de NDIS driver a escolha recaiu sobre o miniport driver, uma vez que é aquele que trabalha directamente com o hardware, neste caso, com o switch.

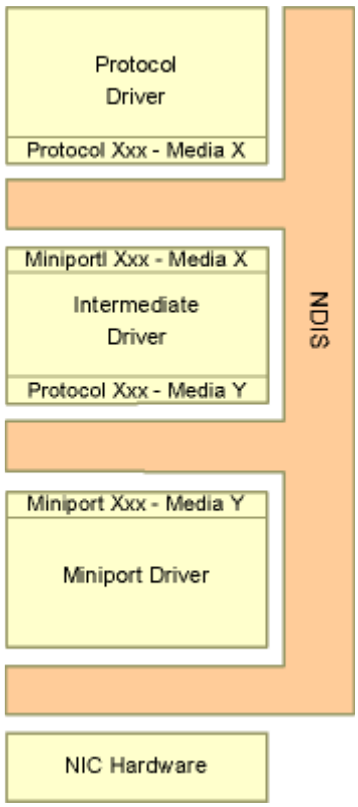


Figura 4.14 - Arquitectura dos drivers NDIS

A parte inferior do driver comunica com o hardware, enquanto a parta superior do driver possui uma interface que permite aos drivers de protocolo receber e enviar pacotes pela rede. Um driver miniport mantém informação sobre a sua capacidade e estado, assim com informação sobre a interface (NIC) que controla. Esta informação é identificada através de OIDs (Object Identifiers) que já se encontram definidos pelo sistema.

O primeiro passo para criar um driver NDIS é declarar o driver no registry, onde é colocada toda a informação importante, desde a dll que deve ser usada para aceder ao driver até ao(s) endereço(s) IP e máscara(s) de rede do NIC.

A partir do momento que se encontra declarado no registry, o OS ao carregar a dll vai aceder ao ponto de entrada do driver, a função DriverEntry, que é obrigatória neste tipo de drivers. Na função DriverEntry é registado o miniport. Este registo consiste em especificar as características do driver através da estrutura NDIS_MINIPORT_CHARACTERISTICS.

```
typedef struct _NDIS_MINIPORT_CHARACTERISTICS {
  UCHAR MajorNdisVersion;
  UCHAR MinorNdisVersion;
  UINT Reserved;
  W_CHECK_FOR_HANG_HANDLER CheckForHangHandler;
  W_DISABLE_INTERRUPT_HANDLER DisableInterruptHandler;
  W_ENABLE_INTERRUPT_HANDLER EnableInterruptHandler;
  W_HALT_HANDLER HaltHandler;
  W_HANDLE_INTERRUPT_HANDLER HandleInterruptHandler;
  W_INITIALIZE_HANDLER InitializeHandler;
  W_ISR_HANDLER ISRHandler;
  W_QUERY_INFORMATION_HANDLER QueryInformationHandler;
  W_RECONFIGURA_HANDLER ReconFiguraHandler;
  W_RESET_HANDLER ResetHandler;
  W_SEND_HANDLER SendHandler;
  W_SET_INFORMATION_HANDLER SetInformationHandler;
  W_TRANSFER_DATA_HANDLER TransferDataHandler;
  W_RETURN_PACKET_HANDLER ReturnPacketHandler;
  W_SEND_PACKETS_HANDLER SendPacketsHandler;
  W_ALLOCATE_COMPLETE_HANDLER AllocateCompleteHandler;
  W_CANCEL_SEND_PACKETS_HANDLER CancelSendPacketsHandler;
  W_MINIPORT_SHUTDOWN_HANDLER AdapterShutdownHandler;
} NDIS_MINIPORT_CHARACTERISTICS,
```

MajorNdisVersion - Especifica a parte superior da versão da livraria NDIS que o driver vai usar.

MinorNdisVersion - Especifica a parte inferior da versão da livraria NDIS que o driver vai usar.

CheckForHangHandler - Especifica o entry point da função MiniportCheckForHang, caso exista. A função MiniportCheckForHang é uma função opcional que reporta o estado da NIC.

DisableInterruptHandler - Especifica o entry point da função MiniportDisableInterrupt, caso exista. A função MiniportDisableInterrupt é opcionalmente implementada pelos drivers de NICs que suportam o activar e desactivar dos interrupts, mas não partilham um IRQ com outros NICs.

EnableInterruptHandler - Especifica o entry point da função MiniportEnableInterrupt, caso exista. A função MiniportEnableInterrupt é opcional, é fornecida por drivers de NICs que suportam a activação e desactivação dinâmica de interrupts, mas não partilham um IRQ com outros NICs..

HaltHandler - Especifica o entry point da função MiniportHalt. A função MiniportHalt é obrigatória, sendo a função que liberta recursos quando a NIC é removida.

HandleInterruptHandler - Especifica o entry point da função MiniportHandleInterrupt. A função MiniportHandleInterrupt é obrigatória se o driver da NIC gerar interrupts, esta função faz o tratamento de todas as operações de interrupt.

InitializeHandler - Especifica o entry point da função MiniportInitialize. A função é obrigatória e prepara a NIC para as operações de rede, reclama no registry todos os recursos necessários para a NIC e reserva os recursos que o driver necessita para realizar as operações de rede.

ISRHandler - Especifica o entry point da função MiniportIsr. Esta função é obrigatória caso o driver da NIC gere interrupts.

QueryInformationHandler - Especifica o entry point da função MiniportQueryInformation. Esta função é obrigatória e retorna informação sobre as capacidades e estado do driver da NIC ou da NIC em si.

ReconFiguraHandler - Especifica o entry point da função MiniportReconFigura. Esta função não é chamada pela livraria NDIS, mas pode ser chamada pela função MiniportInitialize.

ResetHandler - Especifica o entry point da função MiniportReset. A função MiniportReset é uma função obrigatória que ordena um reset de hardware à NIC e faz reset ao software do driver.

SendHandler - Especifica o entry point da função MiniportSend, MiniportWanSEnd ou NULL, caso exista a função MiniportSendPackets. Caso o driver suporte envie múltiplos pacotes, deve colocar este campo a NULL e indicar a função para o campo MiniportSendPackets.

SetInformationHandler - Especifica o entry point da função MiniportSetInformation. Esta é uma função obrigatória que permite aos drivers de protocolo requerer alterações ao estado da informação que o miniport mantém nos seus OIDs particulares, tais como alterações nos endereços multicast.

TransferDataHandler - Especifica o entry point da função MiniportTransferData, caso exista. Esta função é obrigatória, com excepção dos casos em que quem chama a função é um driver de uma WAN ou em que especifica que pode receber múltiplos pacotes.

ReturnPacketHandler - Especifica o entry point da função MiniportReturnPacket ou NULL.

SendPacketsHandler - Especifica o entry point da função MiniportSendPackets. Mesmo que um driver preencha o entry point SendHandler e SendPacketsHandler, a função SendPacketsHandler é a função chamada.

AllocateCompleteHandler - Especifica o entry point da função MiniportAllocateComplete, caso exista. Esta função é opcional, encontra-se disponível em drivers de NICs que são masters de bus DMA e que utilizam NdisMAllocateSharedMemoryAsync.

CancelSendPacketsHandler - Especifica o entry point da função MiniportCancelSendPackets, caso exista. Os drivers miniport que coloquem em filas de espera pacotes por mais de um segundo, devem registar uma função MiniportCancelSendPackets.

PnPEventNotifyHandler - Especifica o entry point da função MiniportPnPEventNotify. Os drivers da versão 5.1 devem registar uma função MiniportPnPEventNotify. A função lida com a notificação de eventos Plug and Play.

AdapterShutdownHandler - Especifica o entry point da função MiniportShutDown, caso exista. Os drivers da versão 5.1 devem registar uma função MiniportShutDown. Esta função coloca uma NIC no seu estado inicial quando o sistema é desligado, quer tenha tido origem no utilizador ou num erro de sistema que tenha ocorrido.

4.4 Device Drivers do EPC

Como já foi referido, o sistema operativo do EPC possui 3 device drivers:

- um device driver para comunicar com o DS2430A
- um device driver para comunicar com o DSP (Digital Signal Processor), RTC (Real Time Clock) e EEPROM (Electrically-Erasable Programmable Read-Only Memory) através de I2C.
- um device driver para comunicar com a FPGA

4.4.1 Device driver da EEPROM DS2430A

O DS2430A é uma EEPROM a 256 bits de 1 fio que identifica e armazena informação relevante acerca do produto ao qual está associada. Este dispositivo contém:

- um número de registo de fábrica gravado a laser, que inclui um número de série de 48 bits, mais 8 bits de CRC e 8 bits de identificação do código da família do dispositivo (0x14).
- 256 bits EEPROM programáveis pelo utilizador
- 64 bits de um registo de aplicação, programável uma vez.

Toda a comunicação com o dispositivo é feita através de um único fio.

O número gravado de fábrica no DS2430A é visto como o número de série do dispositivo, que é único, sendo este número utilizado para identificar os vários ICCs. O valor colocado no registo de aplicação, programável uma vez, corresponde ao endereço MAC da placa ethernet do Triton. Os restantes 256 bits são utilizados para guardar dados à medida que vai sendo necessário.

A comunicação com o DS2430A é feita através de um fio, ligado a um GPIO do processador, permitindo colocar o sinal no fio a um ou zero conforme o desejado.

A dificuldade de fazer um driver para este dispositivo, residiu no facto de os sinais terem de ser mantidos a zero ou a um, durante espaços de tempo na ordem dos microsegundos. No WCE isto coloca algumas dificuldades, porque as funções relacionadas com tempos, mais utilizadas no WCE e Windows em geral(ex: GetTickCount(), sleep(), etc) têm como unidade mínima o milisegundo. Assim sendo e tendo em conta que a comunicação com o DS2430A deve obedecer com rigor a determinados parâmetros de tempo, foi necessário encontrar um meio que permitisse obter valores de tempo em microsegundos. O problema foi resolvido através dos high-resolution timers, que através das funções QueryPerformanceCounter e QueryPerformanceFrequency, permitem obter tempos cuja unidade mínima é inferior a 1 milisegundo. O QueryPerformanceCounter devolve um valor de 64 bits que representa a contagem dos ticks de alta performance, o QueryPerformanceFrequency devolve o número de ticks de alta performance por segundo. O Valor do QueryPerformanceFrequency é usado para obter um tempo a partir do valor de QueryPerformanceCounter.

$$T_{\mu_s} = \frac{(QueryPerformanceCounterEnd - QueryPerformanceCounterIni) \cdot 1000000}{QueryPerformanceCounterFrequency}$$

64-BIT Lasered ROM

Cada DS2430 contém um código único de 64 bits. Os primeiros 8 bits são o código da família do dispositivo, os 48 seguintes são o número série único e os últimos 8 bits correspondem ao CRC dos primeiros 56 bits. O CRC é gerado através de uma função polinomial ($X_8 + X_5 + X_4 + 1$).

64-BIT LASERED ROM Figure 3

8-Bit CRC Code		48-Bit Serial Number		8-Bit Family Code (14H)	
MSB	LSB	MSB	LSB	MSB	LSB

Figura 4.15 – Organização dos 64 bit lasered ROM

1-WIRE CRC GENERATOR Figure 4

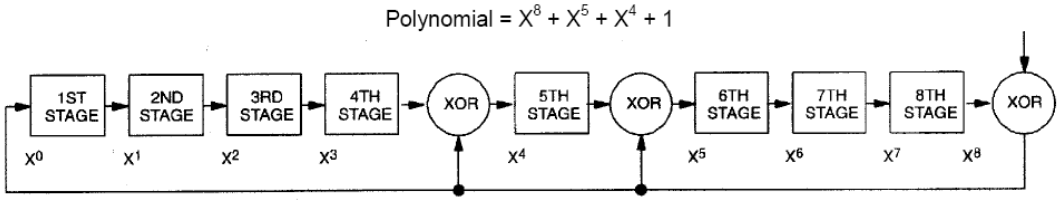


Figura 4.16 – Gerador de CRC

Memória

A memória do DS2430A consiste em três secções separadas, designadas por data memory, application register e status register. A data memory e application register possuem um buffer, utilizado nas escritas ao dispositivo e podem ser escritas e lidas tantas vezes quantas as necessárias. No entanto o application register só pode ser escrito uma vez. Uma vez escrito, passa automaticamente a um estado de protecção contra escritas. O status register indica o estado do application register. Enquanto o application register não for programado, o status register possui um valor de 0xff; uma vez programado este valor passa para 0xfc.

DS2430A MEMORY MAP Figure 5

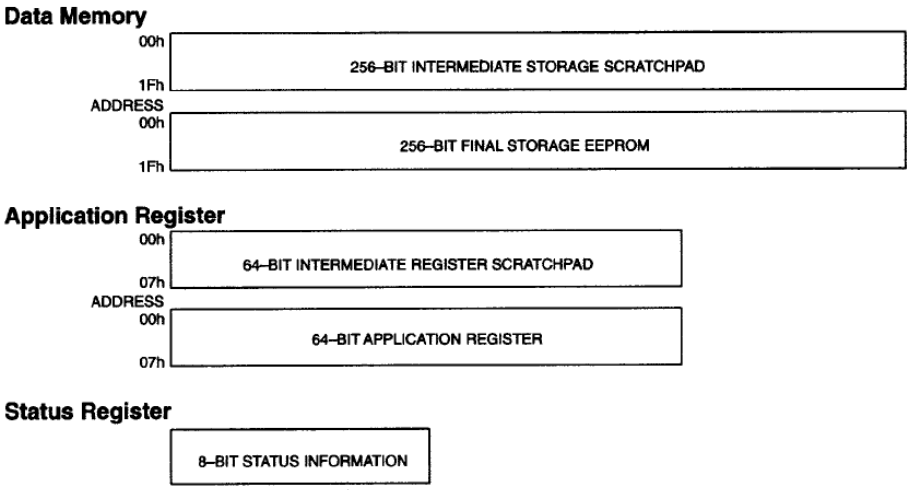


Figura 4.17 – Mapa de memória do DS2430A

Sinalização

O DS2430A requer um protocolo de sinalização rigoroso de modo a assegurar a integridade dos dados. O protocolo consiste em 4 tipos de sinalização: Sequência de reset (com sinalização de reset e sinalização de presença), Escrita0, Escrita1 e leitura de dados. Todos os sinais, com excepção da sinalização de presença, são inicializados pelo master. Para iniciar a comunicação com o DS2430 é necessário uma sinalização de Reset, seguida de sinalização de presença. Esta última indica que o DS2430A se encontra pronto a aceitar comandos.

INITIALIZATION PROCEDURE "RESET AND PRESENCE PULSES" Figure 9

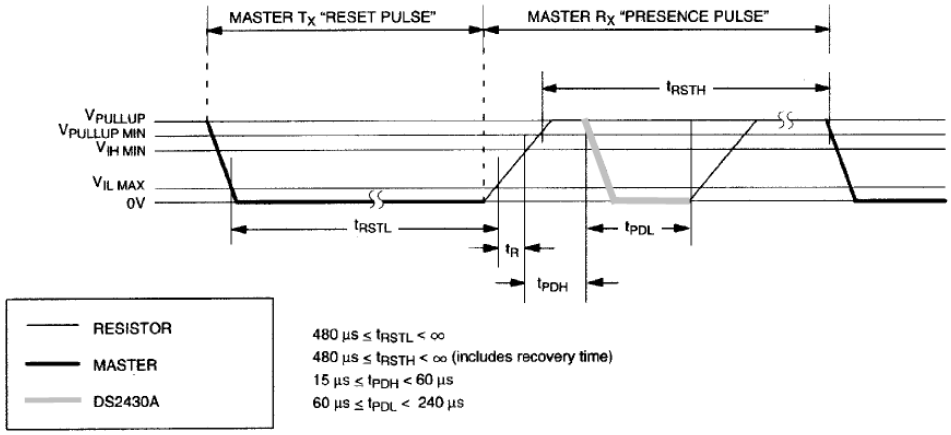


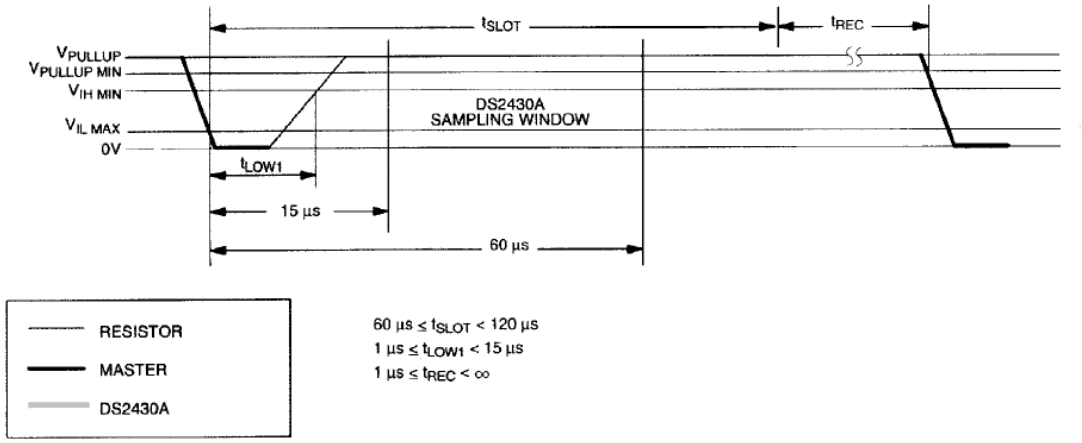
Figura 4.18 – Procedimento de inicialização do DS2430A

Escritas e leituras

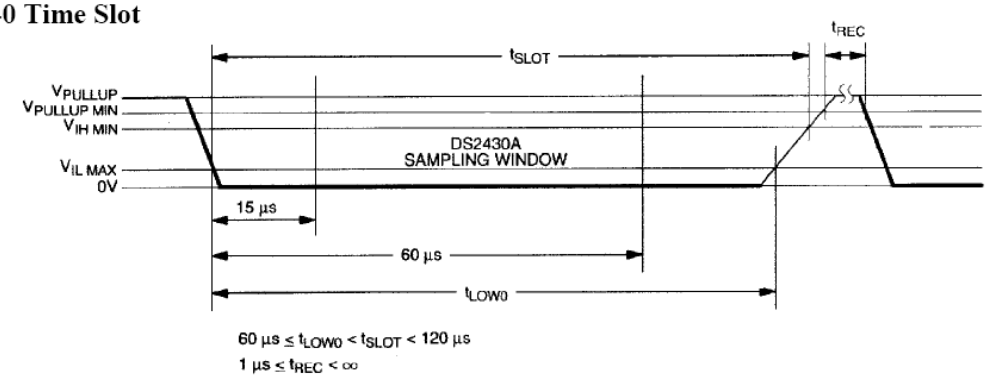
As escritas e leituras são iniciadas com o master a colocar na linha o valor 0. O flanco ascendente da linha de dados, sincroniza o DS2430A com o master. A partir daí o master pode começar a transmitir os dados.

READ/WRITE TIMING DIAGRAM Figure 10

Write-1 Time Slot



Write-0 Time Slot



Read-data Time Slot

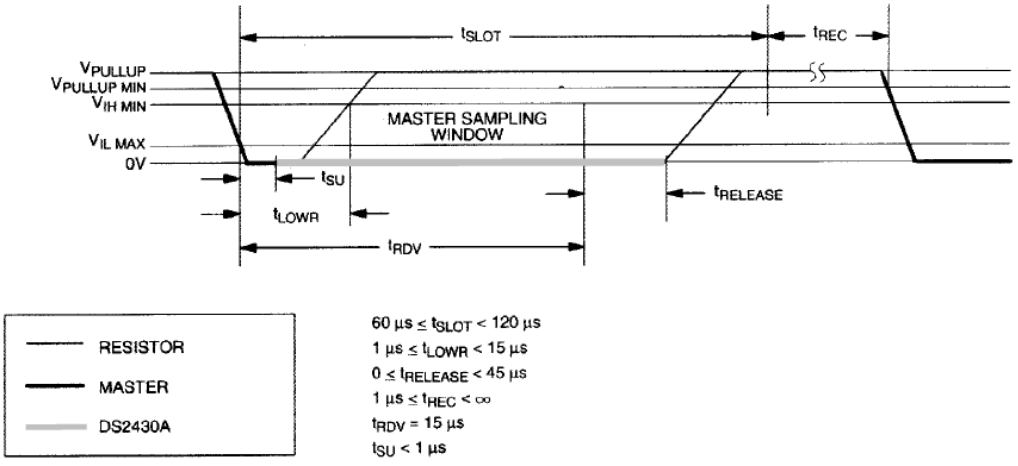


Figura 4.19 – Diagrama de leituras e escritas

4.4.2 Device driver do relógio de tempo real DS1339

O DS1339 é um relógio de tempo real (RTC – Real Time Clock) série. É um relógio/calendário de baixo consumo com dois alarmes programáveis. O endereçamento e transferência de dados são feitos através do bus I2C. O relógio/calendário proporciona informação sobre segundos, minutos, horas, dia, data, mês e ano. A data no final do mês é ajustada automaticamente para meses com menos de 31 dias, incluindo correcções para anos bissextos. O relógio pode funcionar no formato de 24 horas ou 12 horas com indicação de AM/PM. O DS1339 tem um circuito que detecta falhas de energia e comuta automaticamente para a bateria suplente.

Convém salientar que o Triton dispõe de um RTC interno que não é usado devido ao elevado consumo em modo *backup*. Para se obter o consumo mais baixo possível no DS1339, é necessário programá-lo em modo square wave off – saída SQW inactiva.

Modo de funcionamento

O DS1339 funciona como um slave no serial bus. O acesso é feito através do envio de uma condição de Start com a identificação do device, seguido dos dados. Os registos podem ser acedidos até aparecer uma condição de STOP.

Ao escrever ou ler os registos do relógio e calendário, são usados buffers de modo a evitar erros quando os registos internos são actualizados. Nas leituras e escritas os buffers são sincronizados com os registos internos no caso de algum Start ou Stop e quando o ponteiro passa da última posição (10h) para a primeira (00h).

Durante um acesso com de vários bytes (escrita ou leitura) quando o apontador alcançar o fim do espaço de registos (10h), volta para início (00h).

4.4.3 Device driver da EEPROM série 24AA512/24LC512/24FC512

A 24AA512/24LC512/24FC512 é uma EEPROM série de 64K x 8 (512 Kbit). Foi desenvolvido para aplicações avançadas, de baixo consumo, tais como comunicações pessoais e aquisição de dados. Este dispositivo possui a capacidade de escrita de até 128 bytes de cada vez e é capaz de leituras sequências ou aleatórias até ao limite de 512k.

Endereçamento

O byte de controlo é enviado pelo master logo a seguir ao START. O byte de controlo consiste num código de 4 bits. No caso do 24XX512 o valor corresponde a 1010 para operações de escrita e leitura. Os 3 bits seguintes são os de Chip Select. Os bits de Chip Select permitem ter no mesmo bus até 8 dispositivos e são utilizados para seleccionar qual o dispositivo a aceder. O último bit indica o tipo de operação, 1

indica leitura, 0 indica escrita. Os dois bytes seguintes indicam o endereço do primeiro byte de dados.

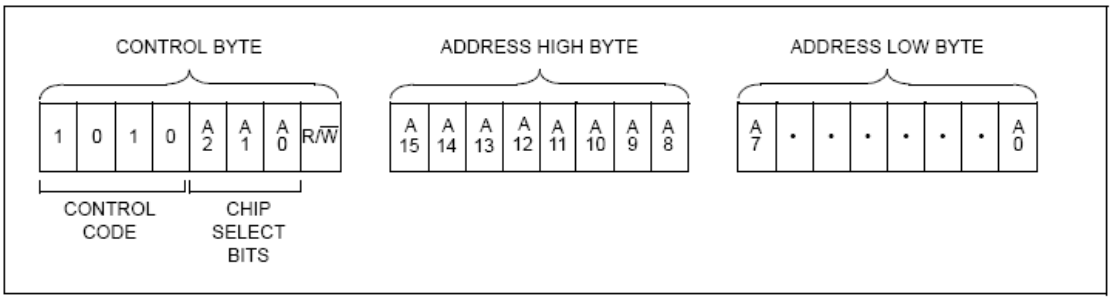


Figura 4.19 – Endereçamento da EEPROM 24XX512

Escrita

A escrita pode ser feita byte a byte, ou pode ser feita a escrita de uma página inteira. Neste caso, o master, em vez de gerar um STOP no final da transferência do byte, continua a transmissão de dados que pode ir até 128 bytes, que são colocados temporariamente num buffer. São escritos na memória após o master enviar uma condição de STOP. Se o master transmitir mais de 128 bytes antes de gerar o STOP o contador de endereços volta ao início apagando os dados enviados anteriormente. O envio do STOP por parte do master gera um ciclo de escrita interno. Durante este ciclo não são gerados acknowledges, o que permite saber quando começa e acaba um ciclo de escrita interna.

A escrita de páginas está limitada à escrita de bytes dentro uma única página física, independentemente do número de bytes escritos. Os limites dos endereços das páginas físicas começam em endereços múltiplos do tamanho do buffer da página e terminam em endereços múltiplos de [tamanho da página - 1]. Se um comando de escrita tentar escrever para além do limite da página física, o contador de posição volta a apontar para o início da página (os dados lá existentes serão rescritos) em vez de passar para a página seguinte.

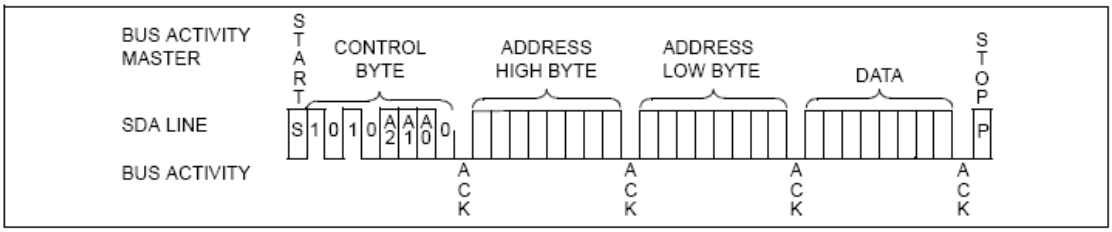


Figura 4.20 – Escrita de um byte

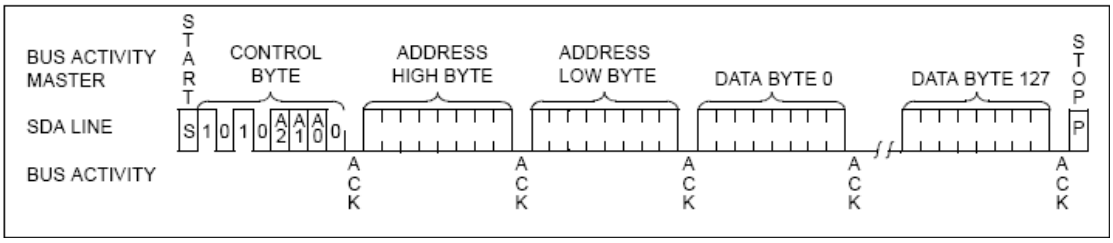


Figura 4.21 – Escrita na sequencial

Leitura

As leituras são iniciadas do mesmo modo, com excepção para o bit que indica que tipo de operação se pretende realizar.

As leituras podem ser de três tipos: endereço corrente, endereço aleatório ou sequencial.

A leitura sequencial e aleatória são semelhante à escrita. O primeiro byte enviado é o byte de controlo, seguido dos bytes de endereçamento e depois os dados. Na leitura de endereço corrente os bytes de endereçamento não são enviados e a leitura é feita a partir da última posição do contador de endereços.

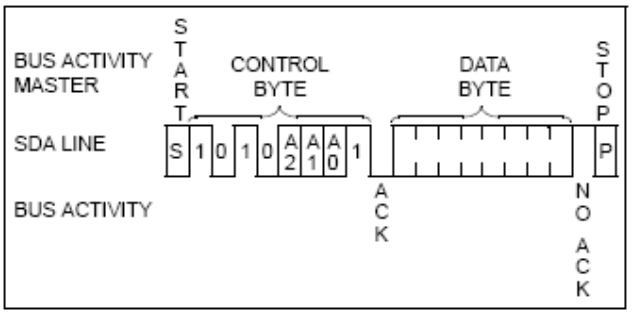


Figura 4.22 – Leitura corrente

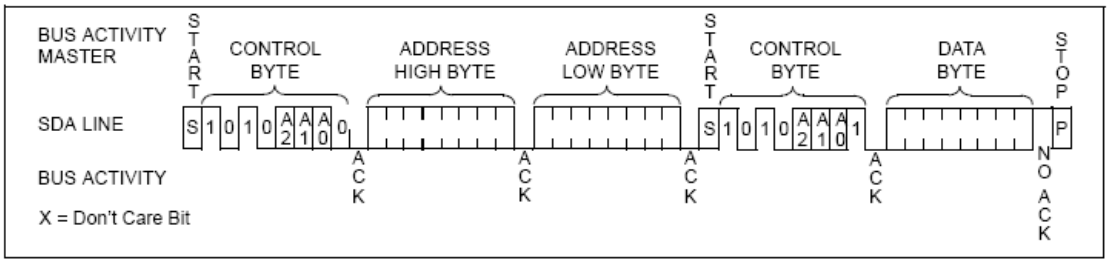


Figura 4.23 – Leitura de endereço aleatório

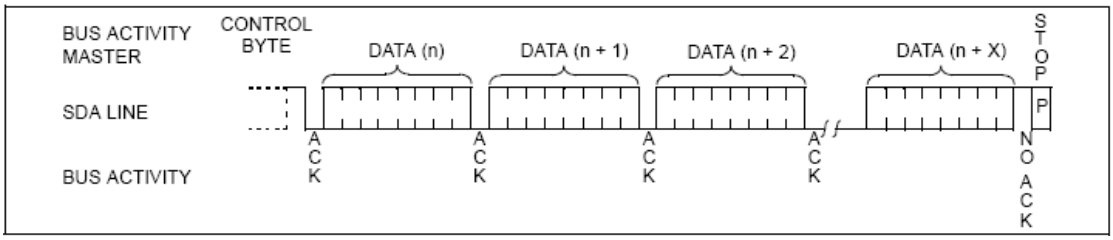


Figura 4.24 – Leitura sequencial

4.4.4 Device driver do processador digital de sinal TMS320VC5509A

O processamento digital de sinal é efectuado pelo DSP TMS320VC5509A. Efectua processamento digital de áudio, ocupando uma posição entre o EPC e a FPGA, disponibilizando serviços a ambos os blocos. Estão previstos os seguintes serviços:

- até 15 canais VAD (voice activity detector para funções de VOX)
- até 16 canais NR (noise reduction)
- até 5 canais de detecção DTMF
- 1 monitor de nível
- até 4 codecs para voip (G711, G723, 1000/1200bps)

As ligações ao EPC são efectuadas através do bus I2C e FPGA. O controlo do DSP é efectuado pelo EPC e consiste na configuração dos serviços de VAD, NR e DTMF e no sentido contrário, das informações dos detectores VAD e DTMF. O áudio comprimido proveniente/destinado aos 4 codecs de voip passa pela FPGA. O tráfego máximo é atingido com os codecs G711 (4x8000x8=256kbps).

Nas comunicações com o DSP, tanto o EPC como o DSP podem ser master ou slave.

O driver I2C do EPC que comunica com o DSP, possui uma imagem dos dados do DSP. Sempre que do lado do DSP existem novos dados é enviada uma mensagem ao driver I2C do EPC, para que este actualize a imagem dos dados que possui. Quando a aplicação que acede ao driver I2C do DSP, pede para fazer uma leitura, não está realmente a efectuar uma leitura ao DSP, mas sim à imagem dos dados do driver. Deste modo o DSP não precisa de armazenar informação e a aplicação perde menos tempo na leitura dos dados, uma vez que não é necessário perder tempo em estabelecer a comunicação com o dispositivo e a ler os dados pretendidos.

Para receber os dados do DSP, foi criada uma thread, no driver I2C do EPC, que está "pendurada" num evento associado a um interrupt do bus I2C. Este interrupt é activado quando o endereço de slave do EPC é detectado numa mensagem enviada pelo DSP. Do lado do driver I2C é então lida a informação enviada pelo DSP e actualizada a sua imagem de dados.

Para activar o interrupt e o evento associado foi necessário fazer algumas alterações em ficheiros chave do WCE. Foi necessário declarar o evento associado ao interrupt no ficheiro oalintr.h.

Neste ficheiro estão declarados os eventos associados aos interrupts de sistema da camada OAL. A camada OAL não é mais do que a camada existente entre o kernel do WCE e o hardware do dispositivo onde corre o WCE. Esta camada facilita a comunicação entre o sistema operativo e o dispositivo alvo. Após terem sido chamadas pelo boot loader, as rotinas OAL implementam a inicialização da plataforma, rotinas de processamento de interrupts, enable/disable interrupts, etc.

Foi também necessário alterar o ficheiro cfwxsc1.c para permitir que os interrupts do bus I2C pudessem ser activados ou desactivados, conforme o driver I2C necessitar. Este ficheiro implementa as interfaces do kernel para interrupts de firmware da placa.

O ficheiro intxsc1.c também foi alterado; este ficheiro é onde está implementada a ISR (Interrupt Service Routine) que trata os interrupts gerados. Por cada interrupt do bus I2C despoletado, a ISR verifica a origem do interrupt, e se o interrupt foi gerado pelo facto do DSP estar a tentar contactar o EPC então o ISR despoleta o evento associado a este interrupt.

4.4.5 Device driver da FPGA Xilinx XS3S400

A FPGA (Field Programmable Gate Array) escolhida para este projecto foi uma Xilinx Spartam3 XS3S400.

Para aceder a esta FGPA foi criado um device driver do lado do EPC. Os acessos à FPGA são feitos através do bus local a 16 bits.

O Windows CE implementa uma gestão de memória virtual paginada, semelhante a outras plataformas Windows, onde cada página tem um tamanho de 4,096 bytes (4 KB). A FPGA está indexada ao Chip Select 2.

Para aceder à FPGA foi ainda necessário programar o registo do processador MSC, que permite configurar memória estática ou Variable Latency I/O, que corresponde aos pares chip-select nCS(1:0), nCS(3:2) e nCS(5:4) respectivamente. Um dos parâmetros importantes a ter em conta na programação é o facto do bus da FPGA trabalhar a 16 bits. Como os processadores ARM apenas fazem transferência entre memórias a 32 bits, foi necessário declarar a FPGA como VLIO, de modo a conseguir usar correctamente o bus de 16 bits da FPGA.

Table 6-21. MSC0/1/2 Bit Definitions (Sheet 1 of 3)

0x4800_0008 0x4800_000C 0x4800_0010	MSC0 MSC1 MSC2	Memory Controller

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RBUFF1/3/5	RRR1/3/5			RDN1/3/5				RDF1/3/5				RBW1/3/5	RT1/3/5			RBUFF0/2/4	RRR0/2/4			RDN0/2/4				RDF0/2/4				RBW0/2/4	RT0/2/4		
Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	*	0	0	0

Bits	Access	Name	Description
15	R/W	RBUFFx	Return Data Buffer vs. Streaming behavior. When slower memory devices are being used in the system (e.g. VLIO, slow SRAM/ROM), this bit must be reset to allow the system to not have to remain idle while all data is being read from the device. By resetting this bit, the system is allowed to process other information. When set, the internal bus may halt while all data is returned from the device. The value of the RBUFF bit does not affect the behavior of the external memory bus. Once a transaction begins on the memory bus, it must be completed before another transaction starts. When Synchronous Static memory devices have been enabled for a given bank, this value will default to Streaming behavior (assuming a faster device). The register bit will still read as 0 (Return Data Buffer) unless it has specifically been programmed to a 1. This cannot be overridden. 0 – Slower device (Return Data Buffer) 1 – Faster device (Streaming behavior)
14:12	R/W	RRRx<2:0>	ROM/SRAM recovery time. Chip select deasserted after a read/write to next chip select (including the same static memory bank) or nSDCS asserted is equal to (RRRx * 2) memclks. This field must be programmed with the maximum of t_{OFF} (divided by 2), write pulse high time (Flash/SRAM), and write recovery before read (Flash).
11:8	R/W	RDNx<3:0>	ROM delay next access Address to data valid for subsequent access to burst ROM or Flash is equal to (RDNx + 1) memclks. nWE assertion for write accesses to SRAM is equal to (RDFx + 1) memclks. The nOE (nPWE) deassert time between each beat of read/write for Variable Latency I/O is equal to (RDNx + 2) memclks. For variable latency I/O, this number must be greater than or equal to 2.

Figura 4.25 – Registo MSC0/1/2 bits 8 a 15

Table 6-21. MSC0/1/2 Bit Definitions (Sheet 2 of 3)

0x4800_0008
0x4800_000C
0x4800_0010

MSC0
MSC1
MSC2

Memory Controller

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RBUFF1/3/5	RRR1/3/5			RDN1/3/5				RDF1/3/5				RBW1/3/5	RT1/3/5			RBUFF0/2/4	RRR0/2/4			RDN0/2/4				RDF0/2/4				RBW0/2/4	RT0/2/4		
Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	*	0	0	0

Bits	Access	Name	Description
7:4	R/W	RDFx<3:0>	ROM delay first access. RDF programmed RDF value interpreted 0-11 0-11 12 13 13 15 14 18 15 23 Address to data valid for the first read access from all devices except VLIO is equal to (RDFx + 2) memclks. Address to data valid for subsequent read accesses to non-burst devices is equal to (RDFx + 1) memclks. nWE assertion for write accesses (which are non-burst) to all Flash is equal to (RDFx + 1) memclks. nOE (nPWE) assert time for each beat of read (write) is equal to (RDFx + 1) memclks for Variable Latency I/O (nCS[5:0]). For Variable Latency I/O, RDFx must be greater than or equal to 3.
3	R/W	RBWx	ROM bus width 0 – 32 bits 1 – 16 bits For reset value for RBW0, see Section 6.8. This value must be programmed with all memory types including Synchronous Static Memory. This value must not change during normal operation.

Figura 4.26 – Registo MSC 0/1/2 – bits 3 a 7

Table 6-21. MSC0/1/2 Bit Definitions (Sheet 3 of 3)

0x4800_0008
0x4800_000C
0x4800_0010

MSC0
MSC1
MSC2

Memory Controller

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	RBUFF1/3/5	RRR1/3/5			RDN1/3/5				RDF1/3/5				RBW1/3/5	RT1/3/5			RBUFF0/2/4	RRR0/2/4			RDN0/2/4				RDF0/2/4				RBW0/2/4	RT0/2/4		
Reset	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	*	0	0	0

Bits	Access	Name	Description
2:0	R/W	RTx<2:0>	ROM type 000 - Nonburst ROM or Flash Memory 001 - SRAM 010 - Burst-of-four ROM or Flash (with non-burst writes) 011 - Burst-of-eight ROM or Flash (with non-burst writes) 100 - Variable Latency I/O (VLIO) 101 - reserved 110 - reserved 111 - reserved Burst refers to the device's timing. When the subsequent reads from the device take less time than the first read from a device, it is referred to as burst timing. The address bits must also be taken into account for burst timing devices. For example, in a burst-of-four device, only the lower two non-byte address bits can change for burst timing. For 32-bit devices, this is MA[3:2]. The address order can go 00, 01, 10, 11 where the reads from 01, 10, and 11, take less time to come out of the device. For burst-of-eight devices, the lower three non-byte address bits can change. Writes to these devices are non-burst.

Figura 4.27 – Registo MSC 0/1/2 – bits 0 a 2

DMA

Para conseguir aumentar as prestações do sistema optou-se por usar DMA (Direct Memory Access) nos acessos à FPGA. Para tal foi necessário configurar alguns registos do processador. Como já foi referido, foi necessário declarar a FPGA como VLIO (Variable Latency I/O), isto porque de outra maneira a FPGA era vista como memória, o que fazia com que as leituras e escritas fossem feitas sempre a 32 bits, como é característico nos ARM. Uma vez que o bus é a 16 bits, transacções de 32 bits não dariam o resultado esperado.

Ao usar o DMA, os endereços de origem ou destino não devem usar os últimos dois bits, ou até mesmo os 3 últimos se os endereço em causa forem endereços de memória externa, esta é uma limitação a ter em conta quando se pretende utilizar o DMA. Tanto o endereço de origem como de destino deve ser de um endereço de memória físico.

Numa situação dita "normal" a aplicação chama o device driver, passa-lhe um apontador para memória virtual, assim, o device deve obter o endereço físico, para onde esse endereço de memória virtual aponta. Numa primeira fase, para fazer essa conversão, utilizou-se a função LockPages, que obtém um endereço físico a partir de um virtual. Neste caso é também preciso verificar se o endereço e o número de bytes que se pretendem ler/escrever obrigam a mudança de página em memória (4096 bytes por página). Se assim for a operação de leitura/escrita deve ser dividida em várias

operações, ou seja, por cada página acedida deve fazer-se uma chamada ao DMA. Outro dos cuidados a ter prende-se com o flush e invalidate da cache. Se a operação DMA for realizada do periférico para memória deve fazer-se um invalidate à cache, se a operação for da memória para um periférico deve fazer-se um flush à cache. Estes procedimentos têm como objectivo evitar que se leiam dados obsoletos presentes na cache em resultado de operações anteriores, ou que os novos dados não sejam lidos por falta de refrescamento da cache.

Com o passar do tempo verificou-se que a utilização da função LockPages trazia alguns problemas. Por vezes, inexplicavelmente, apareciam algumas falhas nos blocos de dados transferidos entre as duas posições de memória. Estas falhas levaram a que fosse necessário encontrar um outro modo de trabalhar com os dois tipos de endereço, virtual e físico.

Depois de analisar os entry points que o device driver disponibiliza e as necessidades da aplicação que faz uso do driver, optou-se por utilizar o entry point DeviceIOControl para resolver o problema. Assim, sempre que a aplicação pretender reservar espaço em memória para ler ou escrever na FPGA, deve chamar a função DeviceIOControl utilizando o código IOCTL_ALLOC_BUFFER e indicando qual o tamanho do buffer e o apontador para a variável que pretende relacionar com o buffer. O DeviceIOControl vai devolver um apontador para uma estrutura, da qual fazem parte os apontadores virtual e físico. Sempre que a aplicação pretender ler ou escrever informação na FPGA deve usar o apontador virtual a nível da aplicação e sempre que chama uma função do driver deve utilizar o apontador físico.

Ao nível do driver, sempre que se chama o entry point DeviceIOControl, o device driver aloca um determinado buffer através da função AllocPhysMem que retorna dois apontadores para o espaço de memória alocado, o apontador físico e o apontador virtual. Tendo em conta que a aplicação nunca vai reservar mais do que 4096 bytes e que o AllocPhysMem devolve sempre endereços que são início de página, deixa de haver a preocupação de verificar se houve mudança de página. Como neste caso se aloca memória sem cache, todas as questões relacionadas com a cache deixam de existir, simplificando assim o código e reduzindo as possibilidades de falhas.

4.5 Device drivers do Encaminhador

Como já foi referido, um dos requisitos exigidos para os ICCs é que se consigam ligar em rede entre eles, e para tal dispõem de 4 portas Lan e 2 portas fibra óptica, estas 6 portas são controladas por um switch, que é controlado e acedido pelo router por meio de um controlador ethernet. Para esse efeito, o OS do router necessita de um driver que permita interagir com o switch (através do controlador), de modo a que se consiga estabelecer uma ligação entre o StackIP do WCE e o switch. O WCE possui um driver específico para estas situações, chamado NDIS driver.

4.5.1 Device driver NDIS

O WCE .NET 4.2 possui suporte do driver NDIS até a versão 5.1, que foi a versão escolhida para implementar este driver.

Uma vez declarados os entry points e feito o registo, a biblioteca acede à função de inicialização do driver (declarada como entry point) onde são feitas todas as inicializações necessárias. Neste entry point há a destacar a indicação de uma das características importantes do driver, que é se é um driver do tipo serialize ou do tipo deserialize. Um driver NDIS serialize implica que quando uma das funções, declaradas nos entry points estiver a ser executada, nenhuma das outras funções dos entry points ou ela mesma, pode se chamada para execução, sem a que se encontra a ser executada tenha terminado. No caso de um driver NDIS deserialize qualquer umas das funções dos entry points pode ser chamada, mesmo que uma função chamada anteriormente não tenha acabado a sua execução. A escolha recaiu sobre drivers deserialize, uma vez que com estes tipo de driver é possível ter melhores prestações, embora sejam exigidos mais cuidados na hora de proteger os dados e as funções de múltiplos acessos.

Na implementação do driver foram usados como mecanismo de protecção a múltiplos acessos, vários NDIS_SPIN_LOCK. Os NDIS_SPIN_LOCK são usados para sincronizar o acesso a recursos partilhados entre as várias funções do driver.

Durante a inicialização o driver é questionado pelos protocolos de nível superior, das suas capacidades, nomeadamente quantos pacotes consegue receber das camadas superiores de cada vez que a função SendPacketsHandler é chamada, qual o tamanho máximo que a NIC consegue enviar, qual o tamanho dos buffers de transmissão/recepção, etc.

4.5.2 Outras funções do driver NDIS

Na Flash do router existem três ficheiros de grande importância para o funcionamento do ICC, um deles indica quais os IPs do ICC, outro o MAC a atribuir ao switch/driver e por último um outro ficheiro que contém a programação do switch. Todos este ficheiros são criados pelo software operacional.

No arranque o driver arranca sem IP associado e em modo cliente DHCP, assim que a inicialização é concluída, o driver lê o ficheiro com os IPs. Neste ficheiro, a primeira linha contém o IP base que o driver deve ter associado, este IP nunca é alterado, nem removido. O ficheiro pode ainda ter até mais 4 linhas de IPs, tantas quantas as possíveis ligações em routing que o ICC pode efectuar. Nessas linhas pode já estar indicado um IP para a porta respectiva, sendo nestes casos o routing considerado estático. Se a linha não tiver IP o routing é considerado dinâmico e o IP só é colocado no ficheiro depois da negociação de IPs entre ICCs. Da primeira vez que o ficheiro de IPs é lido, o driver lê também o ficheiro que indica qual o MAC a atribuir ao driver/switch, em seguida lê o ficheiro que contém a programação do switch. Uma vez lidos todos os ficheiros, o driver executa a função DeviceIoControl , com o parâmetro IOCTL_NDIS_REBIND_ADAPTER. O comando de rebind faz com que os protocolos superiores questionem o driver sobre as suas características para determinar o que mudou (IP, MAC, etc). Antes de executar o rebind o driver altera a programação do switch e o seu MAC e retira o modo cliente DHCP. A partir daqui, sempre que houver ligações de ICCs e for necessário adicionar ou retirar IPs, as funções a utilizar são AddIPAddress e DeleteIPAddress, isto porque o

IOCTL_NDIS_REBIND_ADAPTER obriga a uma paragem por parte das funcionalidades do driver na ordem dos 30/45 segundos e só se justifica a sua utilização para os casos em que o MAC se altera. Para o caso em que se alteram os IPs as funções AddIPAddress e DeleteIPAddress são suficientes e acabam por ser menos pesadas, fazendo com que as funcionalidades do driver sejam afectadas por espaços de tempo desprezáveis.

Sempre que o operador pretender alterar o estado de alguma porta, seja de routing para switching, routing estático para dinâmico ou vice-versa, o ficheiro com a configuração do switch é alterado e pode também ser alterado o ficheiro com os IPs, caso se altere o tipo de routing (estático/dinâmico). Estas alterações são feitas pelo software aplicacional.

Para além destes três ficheiros, existem ainda outros ficheiros que podem ser criados/destruídos pelo driver. Estes ficheiros, quando existem, não possuem conteúdo, são apenas usados para sinalização. O driver deve verificar periodicamente o estado das portas do switch (link estabelecido ou falha no link) e sempre que o estado de alguma delas se altere, o driver deve criar ou apagar um ficheiro conforme o estado actual da porta. Se foi detectada uma ligação numa porta é criado um ficheiro a indicar a porta ligada; se uma porta tinha uma ligação que deixou de existir, o ficheiro que existia a indicar essa ligação é destruído. O ficheiro criado tem o nome da porta (ex: Lan1). O software aplicacional utiliza esta informação para tentar, pelas portas ligadas, comunicar com outros ICCS de modo a negociar os IPs a adicionar, para conseguir colocar todos os ICCs numa mesma rede. Sempre que uma ligação é quebrada, o software aplicacional deve remover o IP respectivo da sua tabela.

Todas as tarefas que são efectuadas periodicamente são controladas por timers. Estes timers são inicializados e configurados durante a inicialização do driver.

4.5.3 Tabelas MAC/IP

Uma vez que existem várias portas, torna-se necessário manter uma tabela com os IPs e MACS relacionados com as várias portas.
Os pacotes recebidos do switch possuem a seguinte informação [20]:

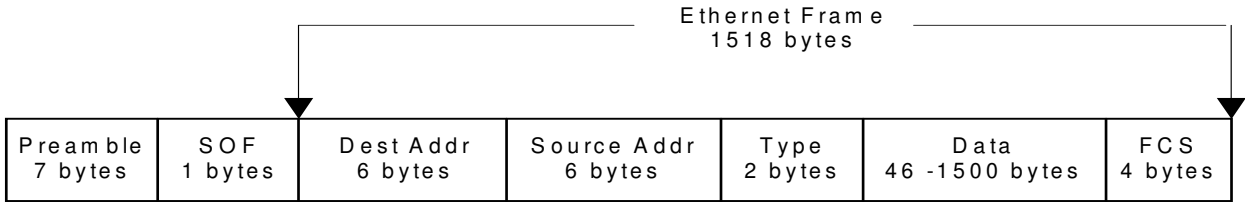


Figura 4.28 – Estrutura de uma frame ethernet

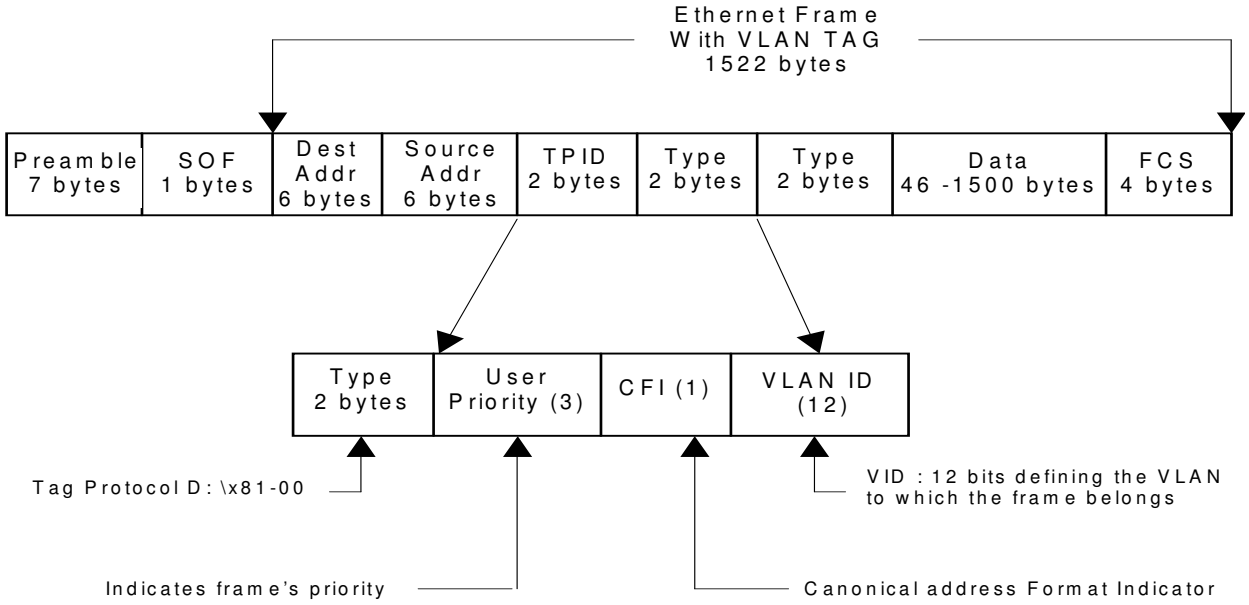


Figura 4.29 – Estrutura de uma frame VLAN

Os 4 bytes adicionados pelo switch permitem identificar a porta por onde chega o pacote. Esses 4 bytes são retirados quando o pacote é passado para o stack. Quando o stack envia para o switch um pacote para ser enviado para a rede, é necessário verificar qual o MAC destino do pacote para obter a porta de saída; uma vez obtida a porta de saída são adicionados os 4 bytes com essa informação, ao pacote recebido do stack, sem essa informação o switch não sabe para que porta deve enviar os pacotes.

A informação com os IPs, Macs e respectivas portas é guardada numa tabela. Esta tabela é actualizada sempre que um pacote ARP ou um pacote IP é recebido através do switch. Se o pacote recebido for um ARP, então é adicionada uma nova entrada à tabela, caso ainda não exista nenhuma com a mesma informação. Se o pacote recebido for um pacote IP, então a tabela é verificada, de modo a que, caso seja necessário, a informação existente seja actualizada. Uma das informações guardadas em cada entrada, é o tempo que decorreu desde que se recebeu o último pacote, com informação relacionada com essa entrada da tabela. De tempos a tempos a tabela é consultada e se este valor for superior a um valor de tempo predefinido, então a entrada é removida.

4.5.4 Negociação de IPs e comunicação com o software operacional

Antes que o protocolo de encaminhamento entre ICCs possa entrar em acção, é necessário que os ICCs adjacentes partilhem uma rede. Para tal é necessário que estes negoceiem os IPs que cada um deve ter, para que todos estejam acessíveis a partir de qualquer ponto da rede. Esta negociação é feita através de um protocolo proprietário. Como a versão do WCE utilizada não permite a utilização de sockets RAW foi necessário encontrar um meio alternativo de fazer com que as mensagens deste protocolo passassem do software operacional para o driver e vice-versa. As messages queues foram o meio encontrado para solucionar este problema.

Inicialmente é criada uma message queue para que o software aplicacional envie mensagens para o driver. Sempre que for detectada uma porta ligada, o software aplicacional indica, através da message queue, que deve ser criada outra message queue, mas desta vez para que o driver envie mensagens para o software aplicacional. Deve ser criada uma message queue por cada porta ligada, ou seja, por cada possível ligação a um ICC. As mensagens do protocolo proprietário de negociacão de IPs, que o software aplicacional pretende enviar para a rede, são entregues ao driver por message queues. Do mesmo modo, as mensagens recebidas pelo driver e que digam respeito ao protocolo proprietário de negociação de IPs, devem ser entregues ao software aplicacional pela message queue correspondente à porta pela qual foram recebidas.

Uma vez estabelecida a comunicação o driver fica ainda encarregue de enviar periodicamente, mensagens que indiquem aos ICCs ligados às suas portas, que a ligação ainda se encontra estabelecida. De igual modo deve receber dos outros ICCs mensagens idênticas.

4.5.5 Múltiplos MACs

Um dos vários dispositivos que se podem ligar a um ICC é um Tactical Media Converter (TMC). Um TMC permite fazer a passagem de dois canais de fibra óptica para dois cabos ethernet. Um TMC não é mais que um switch igual aos que são utilizados pelos ICCs, mas programado de modo diferente. Os dois canais deviam ser vistos pelo TMC de um modo independente, mas quando se começaram a fazer as primeiras experiências não foi isso que se detectou. Ao efectuarem-se os testes com os TMC, concluiu-se que apenas se conseguia comunicar por um dos canais. Após algumas averiguações foi possível concluir que a tabela de ARP do switch do TMC era a mesma para os dois canais (convém salientar que a programação do switch do TMC é bastante diferente da programação do switch dos ICCs) o que criava alguma confusão ao TMC, uma vez que recebia pelos dois canais pacotes com IPs diferentes mas com o mesmo MAC. Para resolver esta situação optou-se por atribuir um MAC a cada porta do ICC, ou seja, cada vez que o stack, que só conhece um MAC, envia algo para a rede, o MAC de origem é alterado conforme a porta por onde é enviado. De igual modo, a todos os pacotes recebidos é alterado o MAC, para o MAC base, de modo a que o stack consiga reconhecer o destino do pacote como sendo o do switch/driver.

4.6 Encaminhamento de pacotes entre ICCs

Como já foi referido, o RIP (versão 1) foi o protocolo de routing escolhido para trabalhar nas rede de ICCs.

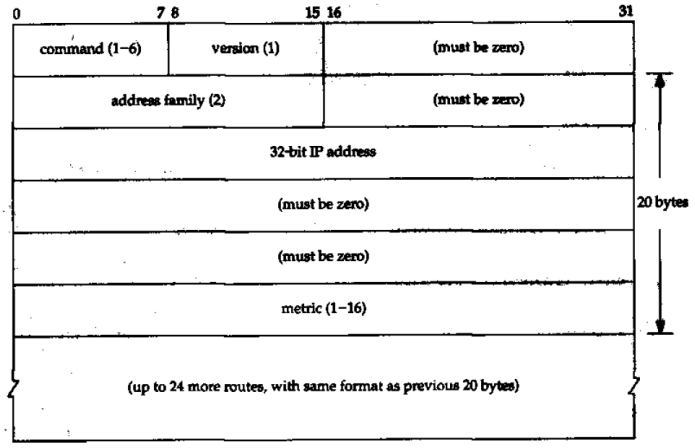


Figura 4.30- Estrutura de um pacote RIP

Como qualquer protocolo de routing, o RIP obriga a que sejam trocadas mensagens entre os routers, de modo a que cada um dos routers tenha a máxima informação possível sobre a rede. É com base nesta informação, que são escolhidos os caminhos a seguir pelos vários pacotes que circulam na rede.

Para implementar o RIP no ICC utilizou-se uma solução semelhante ao GNU Zebra[16]. O GNU zebra é constituído por vários executáveis. Cada protocolo com que o GNU Zebra consegue trabalhar possui um executável associado, ou seja, para trabalhar com o protocolo RIP existe um executável, para trabalhar com o OSPF existe outro executável e por aí em diante. Para além destes executáveis associados aos protocolos existe ainda outro executável que é o responsável por fazer a gestão da tabela de routing e que também faz as distribuições dos caminhos pelos diferentes protocolos de routing.

No caso dos ICCs apenas foram implementados os executável relacionados com RIP e Zebra. A organização do GNU Zebra foi assimilada porque permite deixar uma base de trabalho para projectos futuros, ou seja, se num futuro projecto for necessário usar o RIP basta usar aquilo que já está feito e testado; se for necessário utilizar outro protocolo, basta implementar um executável para esse mesmo protocolo, mantendo o executável que faz a gestão da tabela de routing.

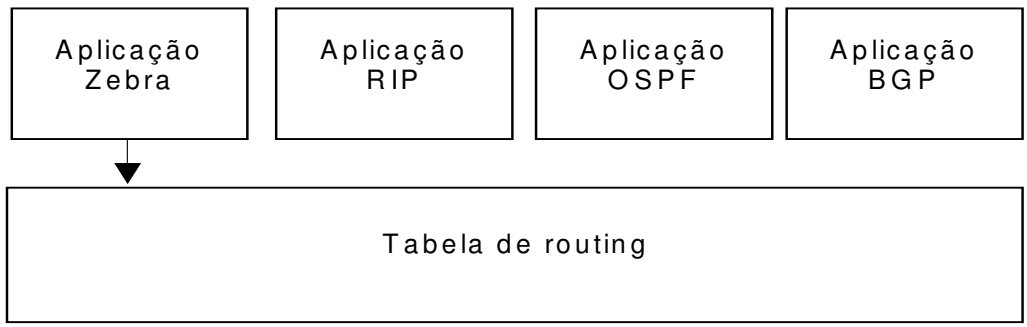


Figura 4.31 – Arquitectura do Zebra GNU

Cada executável possui ainda um ficheiro de configuração. Estes ficheiros são alterados consoante a configuração do sistema, no que diz respeito à rede ethernet, carregada pelo utilizador.

Os comandos de configuração são em tudo semelhantes aos do GNU Zebra.

O sistema operativo encarrega-se de correr estes dois executáveis durante a fase de arranque do sistema

4.7 Suporte de voz sobre IP

Como já foi referido, quando se transmite voz sobre IP, antes de se iniciar a transmissão dos pacotes com os dados relacionados com a voz (designados por pacotes RTP), é necessário existir uma negociação entre os terminais, onde supostamente se trocam alguns dos parâmetros da comunicação, entre os quais os codecs a utilizar durante a comunicação. Os protocolos utilizados para esta negociação são o SIP ou H323.

No que diz respeito aos ICCs, esta negociação não existe, não sendo utilizado nenhum protocolo que faça a negociação entre as duas partes envolvidas na troca de voz sobre IP. A partir do momento em que os ICCs estabelecem uma rede entre si, o Encaminhador sinaliza ao EPC que foi estabelecida uma nova ligação e indica um endereço IP, que corresponde ao endereço IP do EPC que se encontra dentro do ICC no outro lado da ligação. O EPC, ao ver que foi estabelecida uma nova ligação, começa então a enviar pacotes RTP para o endereço que lhe foi fornecido pelo Encaminhador. A partir deste momento e até que a ligação entre os dois ICCs se quebre, estarão sempre a circular pacotes RTP entre os dois ICCS. Existe apenas um codec disponível para o VoIP, no que diz respeito aos ICCs. O codec utilizado foi o G711.

4.8 Stack TCP/IP no Windows CE

O Stack TCP/IP do Windows CE é um stack duplo, isto porque tanto está preparado para IP versão 4 como IP versão 6.

A figura 4.32 apresenta a arquitectura do StackTCP/IP do WCE[17]

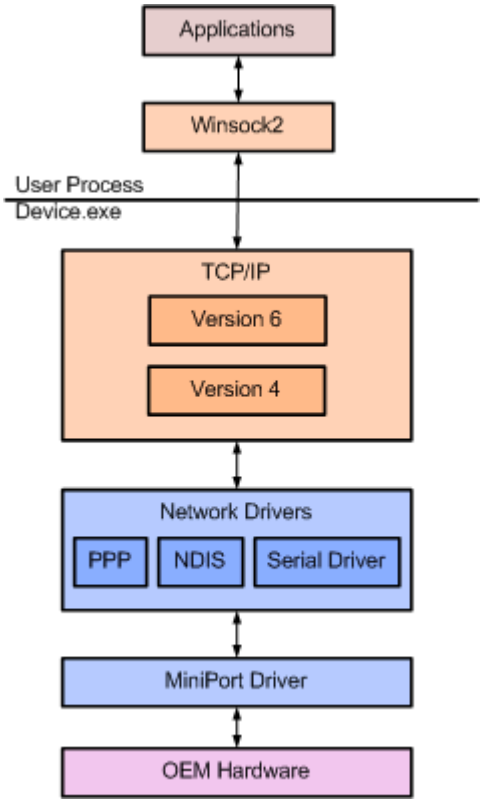


Figura 4.32 – Organização do Stack TCP/IP do WCE

O stack IP do WCE disponibiliza todas as funcionalidades pretendidas, excepto os ao nível do dos sockets (Winsock).

Os sockets do windows permitem aos programadores criar aplicações, que possibilitam a transmissão de dados através da rede, independentemente do protocolo utilizado.

Quando numa aplicação, existe a necessidade de enviar dados pela rede, recorre-se aos sockets. Ao criar um socket, um dos parâmetros que deve ser especificado está relacionado com o tipo de socket que se pretende utilizar. Os tipos mais utilizados são:

SOCK_DGRAM – Um socket datagrama suporta fluxo de dados bidireccional, que não garante que os pacotes de dados sejam entregue numa determinada sequência, que sejam realmente entregues ou que não cheguem em duplicado.

SOCK_STREAM – Um socket stream proporciona fluxo de dados bidirecional, onde é garantido que os pacotes de dados chegam dentro da sequência certa, chegam realmente ao destino e não chegam em duplicado.

SOCK_RAW – Os sockets raw permitem a uma aplicação ter acesso directo a protocolos de comunicação de baixo nível. Os sockets raw são orientados para utilizadores que querem tirar partido de características de protocolos que não estão normalmente acessíveis através de uma interface normal.

Os raw sockets permitem passar dados directamente para as aplicações, passando por cima do processamento TCP/IP, normalmente associado aos pacotes, no encapsulamento/desencapsulamento que o stack TCP/IP realiza.

Inicialmente pensou-se em usar os raw_sockets para determinadas mensagens, que as aplicações a correrem nos ICCs, necessitam de trocar entre si e que não estão associadas a um protocolo existente. Mas após uma consulta à ajuda do Windows CE, chegou-se à conclusão que os sockets raw não foram disponibilizados para esta versão.

Para resolver esta lacuna, optou-se por utilizar message queues. As messages queues permitem a comunicação entre processos. As messages queues disponibilizam um serviço de troca de mensagens entre processos, um processo pode criar uma message queue que pode ser acedida por vários outros processos, estes processos podem ler ou escrever para a message queue. As messages queues têm um funcionamento assíncrono.

Assim sendo criaram-se duas messages queues para a comunicação entre o driver NDIS e a aplicação. Esta message queue servirá para que determinadas mensagens a trocar entre as aplicações dos ICCS e que não estão associadas a nenhum protocolo, possam circular na rede e chegar ao respectivo destino. Uma das message queues é usada pela aplicação para colocar mensagens e para o driver NDIS obter mensagens. A outra message queue é usada pelo NDIS driver para colocar mensagens e pela aplicação para ler mensagens.

O driver NDIS apenas coloca nas messages queues as mensagens que dizem respeito à negociação entre ICCs e que não estão associadas a nenhum protocolo específico, todas as outras seguem o seu caminho normal, ou seja, o stack TCP/IP.

5

Avaliação do trabalho

Neste capítulo compara-se o sistema desenvolvido com outros produtos existentes e apresentam-se os testes realizados ao equipamento.

5.1 Sistemas similares no mercado

Os dois grandes concorrentes de mercado dos ICCS são os sistemas SOTAS do grupo THALES e ROVIS do grupo COBHAM.

5.1.1 SOTAS

O sistema SOTAS é comercializado em três versões, SOTAS, SOTAS M2 e SOTAS IP. A versão SOTAS é a versão base e compreende apenas as comunicações dentro do veículo. As comunicações com o exterior são possíveis apenas por rádio. A versão SOTAS M2 já permite ligações entre veículos e possibilita a utilização de serviços sobre IP, voz e dados. O SOTAS IP, para além das funcionalidades do SOTAS M2, permite ainda a ligação do veículo a WAN, através de fibra óptica ou cabo, onde podem ser usados vários protocolos de routing. Em qualquer uma destas versões a arquitectura utilizada é uma arquitectura em estrela, onde todos os elementos se ligam a uma unidade central (tal como nos ICCS).

5.1.2 ROVIS

Do sistema ROVIS existem também três versões em comercialização ROVIS AN / VIC-3 , ROVIS G2 e ROVIS G2 (TACIP) INTERCOM. A versão AN/VIC-3 continua a ser comercializada pela COBHAM, no entanto, no site da empresa é anunciado que a migração para a versão G2 pode ser feita através da aquisição de alguns módulos, dando a ideia de que a versão ANVIC-3 já se encontra algo ultrapassada. Talvez continue em comercialização para satisfazer mercados menos exigentes em tecnologia de ponta e mais interessados num preço competitivo.

A versão G2 é uma versão optimizada da AN/VIC3, mas com mais algumas funcionalidades extra, como a ligação entre veículos, controlo remoto, etc. A versão G2 TACIP é semelhante à G2, mas permite transferir dados a velocidades superiores (maior largura de banda), permitindo assim o uso de algumas tecnologias multimédia. A arquitectura da ROVIS, ao contrário dos ICCS e SOTAS, não é em estrela. A ligação dos terminais e outros equipamentos ao equipamento central, é feita através de na sua maioria a dois conectores na consola central A esses dois conectores a ligação do equipamento é feita em série.

5.2 Testes

Como acontece com qualquer produto, após as fases de concepção e desenvolvimento, segue-se a fase de testes.

Assim que o ICC sai da linha de montagem deve ser submetido aos testes designados de “testes de produção”, com estes testes pretende-se verificar se todos os componentes do ICC foram bem montados/soldados, se estão correctamente alimentados e se estão preparados para que se comece a carregar o software e firmware, para que a seguir se possam configurar os vários dispositivos.

A primeira fase dos testes de produção, centra-se sobre o hardware, são verificadas as ligações dos vários componentes à placa mãe, as ligações entre os vários dispositivos e as fichas externas, o funcionamento dos LEDs, etc.

Uma vez verificado o hardware, passa-se então ao carregamento do software e firmware. No que diz respeito aos dois subsistemas em estudo, o primeiro passo consiste em carregar para os dois triton o bootloader disponibilizado pelo fabricante. O bootloader pode ser carregado através de um qualquer PC, desde que este possua uma porta paralelo.

O fabricante do Triton, disponibiliza um cabo que permite estabelecer uma ligação entre a porta paralelo do PC e a ficha JTAG do Triton e uma aplicação para PC para fazer download da configuração utilizando esse cabo.

Uma vez carregado o bootloader, é então possível configurar alguns parâmetros dos tritons, como por exemplo o IP, MAC, etc, através do menú que o bootloader disponibiliza via porta série.

Nesta fase serão usadas duas ferramentas para carregar o software. Um cliente FTP e um outro cliente TFTP.

FTP significa File Transfer Protocol, é um protocolo de rede usado para transferir ficheiros de um dispositivo para outro. Utiliza o TCP (Transmission Control Protocol) para as transferências e permite autenticação e navegar entre directorias.

O TFTP é uma versão mais simples do FTP, não permite navegar nas directorias, não possui autenticação de passwords e utilza o UDP (User Datagram Protocol). Este protocolo é tipicamente utilizado para upgrades de firmware de equipamentos de rede.

Com o bootloader carregado e tritons configurados, é então possível carregar o sistema operativo. O sistema operativo é carregado por TFTP, com a ajuda de um cliente TFTP de uso livre.

Depois do sistema operativo carregado e com o Triton a trabalhar, resta colocar as aplicações e configurações através de FTP (File Transfer Protocol).

A partir daqui o ICC encontra-se operacional e a trabalhar com as configurações por defeito, pronto para iniciar os testes de produção. Estes testes são efectuados com os ICCs a trabalhar em modo isolado, no que diz respeito à ethernet.

Os testes de aceitação, são testes efectuados com vários ICCs ligados entre si e com diverso equipamento ligado a cada ICC. O objectivo é testar as várias funcionalidades dos ICCs em conjunto com outros ICCs.

5.2.1 Testes de produção

Os testes de produção pretendem testar o bom funcionamento das várias interfaces dos ICCs.

Durante estes testes, são verificadas todas as interfaces com os rádios, terminais, modems, entre outros.

A interface com a rede também é testada, tanto a interface do EPC como a do router, assim como a interface de rede, situada na placa onde se encontra o controlador ethernet e que é utilizada para operações de debug.

Os leds dos ICCs e terminais são todos verificados, assim como as versões do firmware (FPGA, DSP, etc) e software (Encaminhador, EPC, etc) dos vários dispositivos existentes no ICC.

5.2.2 Testes de aceitação

Os testes de aceitação foram efectuados com 3 ICCs ligados entre si, formando uma rede, que posteriormente era ligada a um router da cisco. Este router da cisco estava ainda ligado à rede ethernet da EID.

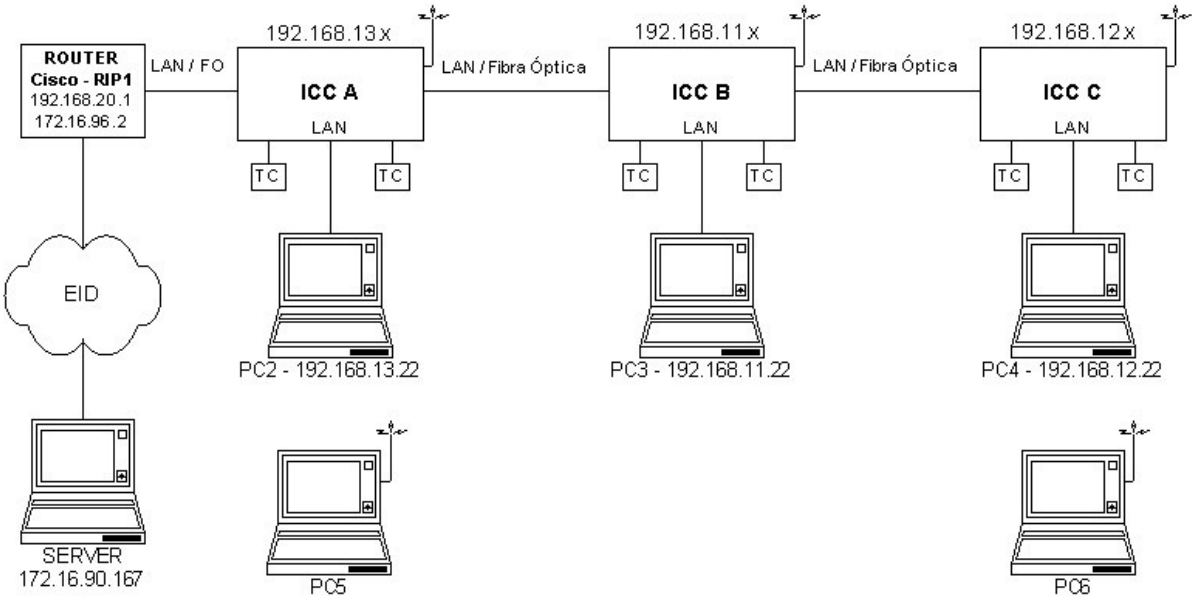


Figura 5.1 – Cenário utilizado para os testes de aceitação dos ICCs

Testes com o Ping

O ping é uma aplicação que permite verificar se um determinado endereço IP existe na rede. Para efectuar essa verificação, é enviada uma mensagem (Echo Request) e esperada uma resposta (Echo Response). Se não chegar uma resposta dentro de um determinado intervalo de tempo, isso significa que não existe nenhum dispositivo com esse endereço, ou caso exista não possui a aplicação Ping.

O que se pretende com este teste é verificar se uma vez ligados os ICCs entre si, estes conseguem formar uma rede. Ou seja, pretende-se verificar se todos os dispositivos da rede estão acessíveis a partir de qualquer ponto dentro da rede.

A rede foi formada apenas com os ICCs, o router da cisco não foi utilizado para este teste.

Este cenário foi mantido durante alguns dias.

Depois de verificar que a rede se mantinha operacional durante vários dias, deu-se início a outro teste, ainda utilizado o Ping. Este outro teste consistia em verificar se ao ligar e desligar algumas das ligações entre ICCs, muitas vezes trocando a porta LAN, ou trocando de LAN para fibra óptica e vice-versa. Com este teste pretendia-se

verificar se a rede recuperava o seu funcionamento depois de falhas ou alterações nas ligações e também quanto tempo demorava a recuperar dessas falhas.

Verificou-se que os ICCs conseguiram formar uma rede e que essa rede conseguia recuperar a alterações nas ligações dos dispositivos que a compõem. O tempo de recuperação a essas alterações revelou-se dentro do esperado, sempre inferior a 1 min.

Teste do acesso à WWW

Depois de verificar que os ICCs tinham capacidade para formar uma rede e que a rede resistia a alterações nas ligações dos seus dispositivos, passou-se ao teste do acesso à WWW.

Para este teste acrescentou-se à rede ICCs um router da Cisco. Este router pretende fazer a ligação entre a rede da EID e a rede ICCs.

Com este teste pretende-se verificar se a rede ICCs ao ser ligada a outra rede consegue continuar a funcionar correctamente, tanto a nível interno, como ao nível de trocas de informação com o exterior.

Utilizado este cenário verificou-se que para além de os ICCS conseguiram manter o funcionamento da rede a nível interno, foi ainda possível aceder à WWW através do router da Cisco

Teste com FTP

O Ping é um bom teste para verificar a existência de determinados endereços IP na rede, mas utiliza pacotes de dados pequenos e muito as mensagens são muitos espaçadas no tempo, ou seja, não permite verificar o comportamento de uma rede quando submetida a esforço.

O objectivo deste teste é manter o cenário do teste anterior, mas acrescentar ainda transferências de ficheiros entre dispositivos. Os ficheiros a transmitir terão de ter um tamanho razoável (50 a 500 MB) e serão transmitidos, sem interrupções, ao longo de vários dias.

O objectivo deste teste é verificar a reacção da rede a uma "sobrecarga" de dados a circular entre os vários dispositivos. A rede deveria manter todas as suas funcionalidades, apesar desta sobrecarga.

Este teste foi mantido durante uma semana, juntamente com o teste do Ping e WWW. No final desse período de tempo verificou-se que a rede se mantinha e que todas as várias transferências de ficheiros tinham sido efectuadas sem falhas. Para além disso a rede continuava a reagir ao teste do Ping e FTP.

Dados como por exemplo a velocidade de transmissão dos ficheiros no teste de FTP, sido propositadamente omitidos por exigência da EID.

6

Conclusões

De um modo geral, os objectivos propostos foram atingidos, tendo os requisitos exigidos pelo cliente foram satisfeitos. De um modo particular, os dois subsistemas descritos nesta tese, apesar de ter alcançado os objectivos pretendidos, pode ainda ser melhorados nalguns aspectos em versões posteriores.

Tanto do lado do EPC como Encaminhador, todas as funcionalidades que se pretendiam para os dois sub-sistemas foram alcançadas. O diver NDIS do Encaminhador terá sido o mais trabalhoso e moroso, devido a sua complexidade.

6.1 Limitações

Existem vários factores, no que diz respeito aos dois subsistemas, que à partida poderão estar a condicionar o funcionamento dos ICCs.

6.1.1 Sistema Operativo Embebido WCE .NET

O sistema operativo embebido WCE .NET. pode estar a limitar as prestações do ICC. Como já foi referido, este OS embebido não permite a utilização de sockets raw. Por esta razão algumas das mensagens trocadas entre os ICCs, são passadas directamente directamente do Device Driver NDIS para as aplicações, através de Message Queues.

Esta solução embora resolva o problema da falta de RAW sockets, faz com que o sistema não consiga obter as mesmas prestações, uma vez que tem de assegurar o processamento das message queues e dos pacotes TCP ao mesmo tempo. Este tipo de sockets (sockets RAW) parece estar disponível nas versões mais recentes do WCE.

Ainda em relação ao WCE .NET, é necessário ter em conta é que o Triton utiliza um processador ARM V5. No entanto o WCE .NET não consegue compilar o código para esta versão do processador, apenas consegue compilar código para os ARM V4. Tendo em conta que o ARM V5 é uma evolução ao ARM V4, é provável que o sistema esteja a sofrer algumas limitações à sua eficiência.

As principais dificuldades encontradas na realização do software prenderam-se com as falhas na documentação da Microsoft, no que diz respeito ao WCE .NET. A Microsoft disponibiliza actualizações frequentes para o WCE .NET, apesar de já terem saído para o mercado duas versões mais recentes. O grande problema é que destas actualizações não fazem parte as actualizações à documentação. A única maneira de conseguir informação actualizada é consultado os fóruns/newsgroups que existem, ou acedendo directamente ao site Microsoft, onde se encontram as últimas versões do Help. Em várias ocasiões nos deparámos com documentação obsoleta ou incompleta, o que leva a que se percam várias horas a tentar resolver problemas que não existiriam se a documentação estivesse correcta e actualizada.

È um facto que a documentação melhorou muito em relação a versões anterior do WCE (ex: versão 2.2 ou 3) onde a documentação era inexistente e onde os fóruns era a única ajuda que o programador possuía.

No entanto este problema de documentação parece ter sido resolvido na versão 6 do WCE. Sempre que se consulta a documentação o programados pode decidir se pretende consultar informação online (onde a documentação está sempre actualizada) ou offline.

6.1.2 Protocolo de encaminhamento RIP

Em relação ao protocolo de encaminhamento utilizado, é sabido que o RIP V1 possui algumas lacunas.

Uma das possíveis soluções para resolver algumas das limitações do RIP V1, seria utilizar o RIP V2, o que poderia ser feito com muito poucas alterações.

As diferenças do RIP V2 para o RIP V1 são essencialmente :

- Realização ou na da autenticação das mensagens.
- Envio das mensagens (Broadcast/Multicast)
- A utilização de Classfull ou ClassLess routing

No RIP V2 é possível autentificar mensagens, o que não acontece no RIP V1. Como é óbvio no que diz respeito à segurança isto significa uma grande evolução.

Também no RIP V2 as mensagens são enviadas em Multicast, enquanto que no RIP V1 são enviadas em Broacast, isto significa que no RIP V1 todas as dispositivos são interrompidas com as mensagens do protocolo, independentemente do interesse que essas mensagens possam ou não ter para o dispositivo. No caso do RIP V2, as mensagens são enviadas apenas para os dispositivos a quem estas mensagens dizem respeito.

O RIP V1 é um protocolo ClassFull enquanto o RIP V2 é um protocolo Classless. Isto significa que o RIP V1, quando divulga as suas redes, não utiliza a máscara associada ao endereço IP. Isto leva a que quando se dimensiona uma rede para trabalhar com RIP V1, os endereços dos dispositivos devem ser escolhidos conforme uma determinada tabela pré-definida. Tudo o que sai fora dessa tabela, não é considerado pelo protocolo. Basta imaginar que na dita tabela o endereço xxx.xxx.xxx.xxx pertence à classe B, ou seja, tem a máscara 255.255.0.0. Se esse mesmo endereço for colocado no dispositivo mas forçado a classe a ser, por exemplo, classe A (mascara 255.0.0.0), quando o RIP divulgar a rede desse IP, divulgará como sendo rede classe B, independentemente daquilo que o dispositivo considerar. Esta limitação não existe no RIP V2, aqui o utilizador é livre de escolher a classe e IP, uma vez que tudo é propagado.

6.1.3 Dispositivo TMC (Tactical Media Converter)

O TMC é um dispositivo desenvolvido dentro do projecto ICCs e que tem como finalidade assegurar a conversão de dois canais de fibra óptica para 2 canais LAN e vice-versa. Um dos componentes do TMC é um switch, igual ao utilizado no ICC, mas com uma programação diferente. O TMC é normalmente ligado às fibras ópticas do ICC.

À partida os dois canais seriam independentes em tudo, mas quando começaram os testes deste dispositivo chegou-se à conclusão que existiam alguns problemas com a tabela de MACs.

Inicialmente o ICC, apesar das suas várias portas, possuía apenas um endereço MAC para todas as portas, embora cada porta tivesse um ip de redes diferentes. Mas quando os testes com os TMCs começaram, chegou-se à conclusão que para o TMC trabalhar correctamente, os dois canais que possui, não podem ter o mesmo MAC, caso contrário a tabela de MACs fica algo “baralhada”.

Para ultrapassar este problema, decidiu-se que para além de ips de redes diferentes, cada porta teria o seu endereço MAC. Estes endereços MAC são definidos

assim que o sistema arranca e não são alterados. Para além dos MACs por canal, existe ainda o MAC base que é o MAC que verdadeiramente corresponde ao driver NDIS e que é aquele que o Stack IP conhece.

Como apenas existia um switch e consequentemente um MAC, foi necessário, em primeiro lugar, colocar o switch em modo promíscuo, ou seja, o switch envia para o driver NDIS todos os pacotes que recebe, independentemente de serem ou não para o seu MAC. É então o driver NDIS que selecciona quais os pacotes que deve passar ao Stack IP. Se o MAC destino da mensagem recebida for para algum dos canais que possui, o driver envia então a mensagem para o stack IP; isto não sem antes ver o seu campo do MAC alterado, o MAC da mensagem passada para o Stack é o MAC base, uma vez que o Stack IP só conhece esse MAC. De igual modo, uma mensagem com origem no stack vê o seu MAC alterado, conforme a porta por onde deve ser enviado.

Em primeiro lugar, o facto de o switch trabalhar em modo promíscuo faz com que o processador esteja constantemente a ser interrompido com a chegada de novas mensagens. Se não estivesse a trabalhar em modo promíscuo, o switch apenas deixaria passar as mensagens em que fosse o real destinatário. Este processo, como é óbvio, consome muito da capacidade de processamento do Encaminhador. Para além disso é também preciso ter em conta o tempo que se perde a alterar os MACs das mensagens enviadas/recebidas.

6.2 Trabalho Futuro

Existem várias perspectivas de trabalho futuro com base nos ICCs.

Um dos possíveis projecto passaria em fazer um género de um ICC mais pequeno e simples, que servisse para equipar outro tipo de veículos mais ligeiros (ex: jipes) que não os carros de combate.

Existem outros possíveis projectos onde se prevê utilizar a montagem do ICC, mas sem router, para aplicações ligadas com o VOIP, onde o ICC serviria de interface entre uma rede rádio e uma rede de telefones H323/SIP.

Bibliografia

1 - http://www.dscc.dla.mil/Programs/MilSpec/ (08/05/2009)
2 - http://www.eid.pt/prdprc525a.htm (08/05/2009)
3 – Qing Li, Caroline Yao, Real Time Concepts for Embedded systems, Elsevier, 2003.
4 – Ed Sutter, Embedded Systems Firmware Demystified, Elsevier, 2002.
5 –Rick Grehan, Robert Moote, Ingo Cyliax, Real-Time Programming, Addison Wesley
6 – Silberschatz, A., e Gagne, Greg, e Galvin, Peter, Operating Systems Concepts, John Wiley & Sons Inc, 2004
7 - CliffordW. Mercer, An introduction to real time operating systems: scheduling teory, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, 1992.
8 – Robert Betz, Introduction to real time operating systems, Department of Electrical and Computer Engineering University of Newcastle, Australia, 2001.
9 - Curso de formação Microsoft, Developing Embedded Solutions for Microsoft Windows CE .NET, 2005.
10- Curso de Redes de Comunicação de Dados - Protocolos de encaminhamento, ISEL, 2007
11 – Michael J. Jipping, Smartphone Operating System Concepts with Symbian OS, John Wiley & Sons Inc, 2007.
12 – Curso de formação Microsoft, Windows CE .NET OAL and Driver Development, 2005.
13 – Deepankar Medhi, KarthiKeyian Ramasamy, Network Routing, Elsevier, 2007.
14 - PXA255 Developers Manual
15 – Douglas Boling, Programming Microsoft Windows CE. .NET, Third Edition, Microsoft Press, 2003
16 - http://www.zebra.org/docs.html (08/05/2009)
17 - http://msdn.microsoft.com/en-us/library/ms881720.aspx (08/05/2009)
18 - http://msdn.microsoft.com/en-us/library/aa447579.aspx (08/05/2009)
19 - http://msdn.microsoft.com/en-us/library/ms892243.aspx (08/05/2009)
20 – Datasheet LAN 91157
21 - *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, 1994, ISBN 0-201-63346-9.