UNIVERSIDADE FERDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

Implementação do Protocolo TCP/IP para Sistemas de Instrumentação

por

ROBERTO SCHÖNHOFEN RIBEIRO

Trabalho de conclusão submetido à avaliação,
como requisito parcial para a obtenção do
grau de Mestre em Informática

Prof. Luigi Carro
Orientador

Porto Alegre, abril de 2002.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Ribeiro, Roberto Schönhofen

Implemetação do Protocolo TCP/IP para Sistemas de Instrumentação / por Roberto Schönhofen Ribeiro. – Porto Alegre: PPGC da UFRGS, 2002.

75 p.: il.+disquete 3,5”

Trabalho de Conclusão (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós Graduação em Computação, Porto Alegre, BR – RS, 2002. Orientador: Carro, Luigi

1.TCP/IP.2.FPGA.3.SASHIMI.4.ASIP.I.Carro, Luigi.II.Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Reitora: Profa. Wrana Panizzi
Pró-Reitor de Ensino: Prof. José Carlos Ferraz Hennemann
Pró-Reitor Adjunto de Pós-Graduação: Prof. Jaime Evaldo Fensterseifer
Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux
Coordenador do PPGC: Prof. Carlos Alberto Heuser
Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

Agradecimentos

Agradeço a Companhia Estadual de Energia Elétrica, na pessoa do Eng. Roberto Silva Dias, então chefe da Divisão Técnica da Distribuição, pela boa vontade demostrada, bem como pelo auxílio prestado, quando da minha intenção de desenvolver o presente trabalho. Agradeço também ao professor Dr. Luigi Carro da Universidade Federal do Rio Grande do Sul, meu orientador, sem o qual certamente este trabalho não teria a qualidade que apresenta. Por fim agradeço a valiosa contribuição prestada pelo MSc. Júlio Carlos Balzano de Mattos, com seu trabalho e dicas importantes, durante a fase de desenvolvimento prático do presente trabalho.

Sumário

Lista de Abreviaturas ... 6
Lista de Figuras ... 8
Lista de Tabelas ... 9
Resumo ... 10
Abstract ... 11
1 Introdução ... 12
1.1 O Estágio Atual da Pesquisa Mundial ... 14
1.2 O Ambiente de Aplicação ... 15
1.3 Enfoque do Trabalho ... 18
1.4 Organização do Trabalho ... 20
2 O TCP/IP Implementado em C ... 21
2.1 O Padrão NE2000 ... 22
2.1.1 Interface de Rede ... 23
2.1.2 Microprocessador Interno ... 23
2.1.3 Memória Local ... 24
2.1.4 Portas de E/S e Registradores ... 25
2.2 O Código Pingador ... 31
2.2.1 O Utilitário PING ... 32
2.2.2 O Hardware de Rede ... 33
2.2.3 Recepção de Dados da Rede ... 35
2.2.4 Envio de Dados para a Rede ... 36
2.2.5 O Protocolo ... 37
2.2.6 Soma de Verificação ... 38
2.2.7 A Resposta ARP ... 38
2.2.8 A resposta ICMP ... 38
2.3 Funções Auxiliares ... 39
2.3.1 O Cálculo de Soma de Verificação ... 39
2.3.2 Função de Inversão MSB – LSB ... 40
2.3.3 Função de Temporização ... 41
2.3.4 Função de Soma de 32 Bits ... 42
2.4 O Padrão Ethernet ... 43
2.4.1 Ethernet no CI Controlador de Rede ... 44

2.4.2 A Implementação do Protocolo 45
2.5 O Protocolo ARP (Address Resolution Protocol) 46
2.5.1 A Resposta ARP 46
2.5.2 A gerência do cache ARP 48
2.6 O Protocolo IP (Internet Protocol) 50
2.6.1 As Simplificações Utilizadas 51
2.6.2 A Implementação do Protocolo IP 52
2.6.3 O Protocolo ICMP (Internet Control Message Protocol) 53
2.6.4 A Implementação do Protocolo ICMP 54
2.7 O Protocolo TCP (Transmission Control Protocol) 55
2.7.1 Simplificações Utilizadas no TCP 56
2.7.2 A Máquina de Estados TCP 57
2.7.3 O Código TCP 58
3 A Geração de um ASIP 60
3.1 A Opção por um Processador Dedicado 60
3.2 A Ferramenta SASHIMI 62
3.3 A Implementação 64
4 Conclusão 70
4.1 Desenvolvimentos Futuros 71
Anexo programa fonte em C (em disquete)
Bibliografia 73

Lista de Abreviaturas

API	application programming interface
ARP	address resolution protocol
ASIP	application specific instruction set processor
AT	advanced technology
BIOS	basic input output system
CEEE	Companhia Estadual de Energia Elétrica
CI	circuito integrado
CRC	ciclic redundant check
CSMA/CD	carrier sense multiple access / collision detection
DMA	direct memory access
DOS	disc operating system
DSP	digital signal processor
E2PROM	electricaly erasable programmable read only memory
E/S	entrada / saída
FIFO	first in first out
FPGA	field programmable gate array
FTP	file transfer protocol
GE	General Electric
HTML	hiper text markup language
HTTP	hiper text transfer protocol
Hz	Hertz
ICMP	internet control message protocol
IEC	International Electrotechnical Comission
IEEE	Institute of Electrical and Electronics Engeneers
IP	internet protocol
IRQ	interrupt request
ISO	International Organization for Standartization
ISA	Industry Standard Architecture
LAN	local area network
LSB	least significant byte
MAC	media access control
MSB	most significant byte
MTU	maximum transfer unit

OSI	open systems interconnection
PC	personal computer
PING	packet internet groper
POP	post office protocol
PPP	point to point protocol
RAM	random access memory
RARP	reverse address resolution protocol
RISC	reduced instruction set computing
RFC	request for comments
SASHIMI	system as software and hardware in microcontrolers
SFD	start of frame delimiter
SMTP	simple mail transfer protocol
TCP	transmission control protocol
UART	universal assincronous receiver transmiter
UDP	user datagram protocol
VHDL	VHSIC hardware description language
WAN	wide area network

Lista de Figuras

FIGURA 1.1 – Sistema de comunicação com TCP/IP. 13
FIGURA 1.2 – Protocolos do centro de operação. 17
FIGURA 1.3 – Encapsulamento de protocolos assíncronos. 18
FIGURA 2.1 - Comparação entre os modelos OSI e TCP/IP. 21
FIGURA 2.2 – Arquitetura da interface de rede. 23
FIGURA 2.3 – Dados adicionados e removidos do quadro. 23
FIGURA 2.4 – Memória local da interface de rede. 24
FIGURA 2.5 – Portas de E/S da interface de rede. 25
FIGURA 2.6 - Formato da mensagem de eco (ICMP). 33
FIGURA 2.7 – Fluxo da rotina de tratamento de interrupções. 34
FIGURA 2.8 – Fluxo da rotina BuscaPct(). 35
FIGURA 2.9 – Fluxo da rotina ExistePct(). 36
FIGURA 2.10 – Fluxo da rotina EnviaPct(). 37
FIGURA 2.11 – Fluxo da rotina de cálculo de soma de verificação. 40
FIGURA 2.12 – Fluxo da rotina de temporização. 42
FIGURA 2.13 - Cabeçalho do quadro Ethernet 43
FIGURA 2.14 – Fluxo do protocolo Ethernet 45
FIGURA 2.15 - Formato dos dados dos protocolos ARP. 47
FIGURA 2.16 – Fluxo da função de resposta ARP. 48
FIGURA 2.17 – Fluxo da função de procura na tabela ARP. 49
FIGURA 2.18 - Formato do datagrama IP 50
FIGURA 2.19 – O fluxograma da rotina IP 52
FIGURA 2.20 - Formato da mensagem ICMP 53
FIGURA 2.21 – Fluxograma da rotina ICMP 54
FIGURA 2.22 - Formato do cabeçalho do TCP 55
FIGURA 2.23 – Máquina de estados simplificada. 57
FIGURA 2.24 – Fluxo do protocolo TCP. 59
FIGURA 3.1 – Arquitetura do microcontrolador FemtoJava. 63
FIGURA 3.2 – Ambiente de teste. 64
FIGURA 3.3 – Troca de dados entre cliente e servidor. 65
FIGURA 3.4 – Fluxo de síntese do microcontrolador TCP/IP. 68

Lista de Tabelas

TABELA 2.1 – Registradores de configuração, página 0. 25
TABELA 2.2 – Registradores de configuração, página 1. 26
TABELA 2.3 – Registradores de configuração, página 2. 26
TABELA 2.4 – Configuração do registrador CR. 27
TABELA 2.5 – Configuração do registrador ISR. 27
TABELA 2.6 – Configuração do registrador IMR. 28
TABELA 2.7 – Configuração do registrador DCR. 28
TABELA 2.8 – Configuração do registrador TCR. 29
TABELA 2.9 – Configuração do registrador TSR. 29
TABELA 2.10 – Configuração do registrador TCR. 29
TABELA 2.11 – Configuração do registrador TSR. 30
TABELA 2.12 - Significado dos indicadores no cabeçalho TCP 56
TABELA 3.1 – Conjunto de instruções implementadas no SASHIMI 64
TABELA 3.2 – Comparação de códigos fonte. 66
TABELA 3.3 – Comparação de códigos executáveis. 67
TABELA 3.4 – Resultados da síntese do hardware 68

Resumo

Baseado na tecnologia de interligação de redes, este trabalho apresenta uma proposta de conexão de dois sistemas com processamento próprio com o intuito de troca de informações, utilizando a pilha de protocolos TCP/IP.

Este sistema será empregado em ambientes de controle industrial, permitindo o envio de informações do servidor de dados para o cliente. Os dados são constituídos de leituras feitas em equipamentos de campo, apresentando ao cliente remoto, medições dos mais diversos tipos. Por outro lado, o cliente poderá enviar comandos aos equipamentos de campo visando o telecontrole.

Como ponto de partida para a elaboração do trabalho prático, foi utilizado o ambiente de controle do sistema de potência da companhia energética do estado do Rio Grande do Sul (CEEE). Um microcomputador com um browser acessa, através de uma rede local, os equipamentos controlados, que poderão ser qualquer tipo de equipamento de campo empregado em subestações de energia elétrica, como disjuntores, transformadores ou chaves.

Para permitir o acesso remoto de tais equipamentos, foi elaborado um servidor de dados constituído de um controlador de rede do tipo Ethernet e um microcontrolador de aplicação específica que se encarrega do processamento da pilha de protocolos. O controlador Ethernet utilizado é um circuito integrado dedicado comercial, que executa o tratamento dos sinais de nível físico e de enlace de dados conforme o padrão IEEE 802.2. O processador TCP/IP, enfoque principal deste trabalho, foi elaborado através da linguagem de programação C, e a seguir traduzido para o Java, que é o ponto de partida para a ferramenta SASHIMI, de geração da descrição em VHDL do microcontrolador de aplicação específica utilizado. O processador TCP/IP encarrega-se da aquisição de dados do equipamento de campo, do processamento da pilha de protocolos TCP/IP, e do gerenciamento do controlador Ethernet.

A partir desta descrição VHDL, foi sintetizado o hardware do microcontrolador em um FPGA, que juntamente com o software aplicativo, também fornecido pela ferramenta utilizada, desempenha o papel de processador TCP/IP para o sistema proposto.

Neste ambiente, então, o cliente localizado no centro de operação, acessa através de um browser o equipamento de campo, visando obter suas medições, bem como enviar comandos, destacando o aspecto bidirecional para a troca de dados e a facilidade de conexão de dois sistemas heterogêneos.

Este sistema pretende apresentar baixo custo de aquisição e de instalação, facilidade de interconexão local ou remota e transparência ao usuário final.

Palavras Chave: TCP/IP, FPGA, SASHIMI, ASIP.

TITLE: "TCP/IP PROTOCOL IMPLEMENTATION TO INSTRUMENTATION SYSTEMS"

Abstract

Based on networks interconnection technologies, this work presents a proposal of connection between two processing systems, aiming at information exchange employing the TCP/IP protocol stack.

This system will be applied in the industrial control environments, allowing the information exchange between the data server and the client. The data is composed of readings made from field equipment, presenting to the remote client several kinds of measurements. On the other hand, the client can send commands to the field equipment implementing remote control.

As a starting point to the elaboration of this work, the control environment of the Power Company of Rio Grande do Sul (CEEE) was utilized. A microcomputer is provided with browser access, through the local network, to the controlled equipment, which can be any kind of field equipment applied in power stations, as circuit breakers, transformers or switches.

In order to allow the remote access of this equipment, a data server build with an Ethernet network controller and an application specific microcontroller was developed. The microcontroller does all the processing of the protocol stack. The Ethernet controller employed is a commercial dedicated integrated circuit, which executes the signal treatment on the physical and data link levels as the IEEE 802.2 standard. The TCP/IP processor was the main goal of this work, and was developed with the C language, and next translated to Java, which is the starting point to the VHDL generation tool SASHIMI, to describe the used application specific microcontroller. The TCP/IP processor is responsible for all data acquisition of field equipment, as well as the TCP/IP stack processing and Ethernet controller management.

From the VHDL output description provided by the SASHIMI tool, the hardware of the dedicated microcontroller was synthesized in a FPGA, with its application software, also supplied by the tool, which plays the role of TCP/IP processor to the proposed system.

In this environment, the client is placed in the operation center, and has access to the field equipment through the browser. This way, it can check the status and send commands, showing the two way path and the easiness of connection between two heterogeneous systems.

This system is intended to present low cost of acquisition and installation, easy interconnection between local and remote equipment and transparency of use to the end user.

Keywords: TCP/IP, FPGA, SASHIMI, ASIP.

1 Introdução

A necessidade de comunicação está pressionando a pesquisa, cada vez mais, a elaborar soluções simples, econômicas e principalmente eficientes, de forma a satisfazer uma necessidade básica do ser humano: a troca de idéias com seus semelhantes de uma forma abrangente e o mais natural possível, ou seja, sem os obstáculos muitas vezes impostos pela complexidade tecnológica. O desenvolvimento destas soluções atualmente caminha a passos largos para o emprego de meios de troca de informação que, de uma maneira ou de outra, atinjam praticamente toda a população mundial, levando em um curto espaço a não se conceber mais uma situação ou local onde os meios de troca de informação com o mundo exterior sejam precários ou mesmo inexistentes.

Neste contexto de exigências do mercado de troca de informações, surge no final da década de 70 uma rede de computadores, que, inicialmente experimental, tornou-se rapidamente um padrão mundial para a troca de dados e mensagens. Essa rede, conhecida como *Arpanet*, foi a princípio fortemente incentivada e financiada pelo governo dos Estados Unidos da América do Norte, e então compartilhada entre instituições militares americanas e algumas universidades. Ao longo dos anos, o custo dos computadores pessoais foi fortemente reduzido, e as conexões a esta rede em muito facilitada ao usuário comercial e particular, produzindo a explosão da mesma pelo mundo afora, quando já era conhecida como *Internet* [TAN 97]. Uma exigência é feita para a interligação na rede mundial: a utilização da pilha de protocolos universalmente padronizada para acesso, o *Transmission Control Protocol / Internet Protocol* (TCP/IP).

Um dos motivos pelo qual a *Internet* fez tanto sucesso em detrimento de tantos outros padrões então existentes é pelo fato da tecnologia empregada ter pelo menos dez anos de testes e aperfeiçoamentos. Vindo ao encontro desta vantagem, de acordo com Douglas E. Commer [COM 98]: "Utilize os padrões dos protocolos existentes sempre que estes padrões se aplicarem; crie novos padrões apenas quando os padrões existentes forem insuficientes e esteja apto para utilizar novos padrões quando eles se tornarem disponíveis e proporcionarem uma funcionalidade equivalente". Pelo até aqui exposto, fica flagrante que qualquer tentativa de conexão para troca de informação entre duas ou mais entidades quaisquer, atualmente, deverá passar por uma análise séria do ponto de vista técnico – econômico, convergindo para a utilização, de alguma forma, dos padrões já amplamente aceitos e disponíveis.

Muitos aspectos técnicos também levam o projetista a sempre avaliar com seriedade a possibilidade de utilização da pilha TCP/IP em novos projetos, e um item fundamental é a maturidade fornecida pelos protocolos, onde anos de pesquisa e desenvolvimento forneceram ao usuário uma pilha bastante funcional e estável do ponto de vista do roteamento de pacotes de dados, desde pequenas distâncias até a distâncias continentais. Para qualquer outra tecnologia que se considere em implementar, os problemas de roteamento, entre outros, a serem enfrentados já foram há muito considerados e

resolvidos no TCP/IP, no grande laboratório de testes da *Internet*, tornando-o uma solução bastante atraente.

A proposta do trabalho aqui apresentado, é viabilizar a conexão de dois sistemas heterogêneos utilizando a pilha de protocolos do TCP/IP, baseado na infra-estrutura já padronizada e bastante difundida das redes locais e globais (LAN – *Local Area Network*, WAN – Wide Area Network). Para atingir tal objetivo foi desenvolvido um hardware específico que se encarrega do estabelecimento da conexão via rede e da troca de dados entre os sistemas cliente e servidor. O hardware é um processador para aplicação específica que possui a qualidade de gerenciar um controlador de rede do padrão Ethernet e processar os pacotes que chegam da rede, gerando uma resposta adequada, segundo a pilha de protocolos. Este processador foi implementado com a utilização de um FPGA (Field Programmable Gate Array). Outra característica importante deste processador é a sua integração, uma vez que todo o gerenciamento do protocolo será diretamente inserido no hardware, liberando recursos de memória e processamento. A figura 1.1 a seguir ilustra de forma simplificada o sistema proposto.

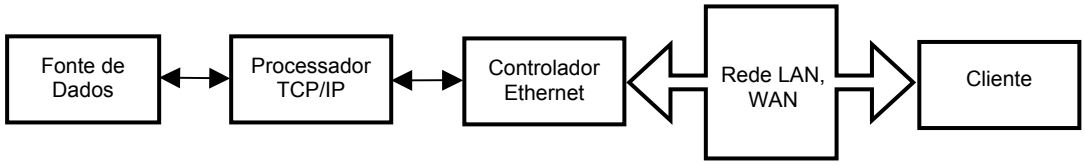

FIGURA 1.1 – Sistema de comunicação com TCP/IP.

Uma aplicação do sistema é a medição, supervisão e controle do sistema de potência atualmente sendo implantado na companhia energética do estado (CEEE – Companhia Estadual de Energia Elétrica). Neste caso, a fonte de dados será a medição de um determinado equipamento de campo de uma subestação de energia elétrica, que possui um conjunto completo de medidas disponível para o usuário, como por exemplo tensão e corrente trifásica, e todas as medidas derivadas destas seis medidas primárias, por exemplo, potência ativa, aparente e reativa, fator de potência, energia, entre outras. Como fonte de dados podemos colocar também o estado dos diversos equipamentos de campo que constituem a subestação como disjuntores, chaves e transformadores, e estes estados são definidos como aberto ou fechado, ligado ou desligado e assim por diante. Acrescente-se a isto o comando destes equipamentos, que em última análise são dados que fluem no sentido inverso dos anteriormente mencionados.

Neste caso específico, o cliente é um computador qualquer executando um sistema operacional como o Windows ou Linux, que possua um navegador Internet com as facilidades do protocolo TCP/IP. Ligado à Internet ou a sua Intranet, Conecta-se ao sistema proposto, que estará localizado diretamente junto ao módulo em questão ou equipamento de campo. De acordo com o grau de segurança ou alcance que se queira atingir será escolhida a rede adequada. Como regra geral, se a segurança é fator determinante, então uma rede privada e dedicada deverá ser utilizada (Intranet). Se, por outro lado, o alcance dos dados é mais importante então a Internet é a escolha adequada.

Neste ponto um controlador Ethernet extrai da rede as informações destinadas ao sistema, que foram originalmente enviadas pelo cliente, e as remete ao processador TCP/IP, que as processa e dá o destino adequado. Se existirem dados que devam ser

enviados ao equipamento de campo, estes serão formatados de maneira adequada e em seguida remetidos. Esta formatação, simplificadamente, poderá ser a serial assíncrona, a qual é amplamente utilizada em sistemas mais simples.

De forma semelhante, os dados que devam ser enviados do campo para o cliente percorrem o sistema no sentido inverso. O processador TCP/IP se encarrega de todos os processos que são necessários antes e durante a conexão fim-a-fim entre cliente e servidor de dados. Esta conexão é estabelecida no nível de transporte de dados com o protocolo TCP, conforme será visto mais adiante, e apresenta uma série de processos de troca de pacotes entre cliente e servidor, antes de se estabelecer a fase de troca de dados propriamente dita.

A partir do ponto de estabelecimento da conexão os dados fluem em ambos sentidos, facilitando o envio de dados do servidor para o cliente, bem como o envio de requisições ou comandos do cliente para o servidor.

Com a intenção de alocar o sistema TCP/IP junto aos equipamentos de campo, e portanto sujeito às intempéries, foi proposto que o resultado final deve ser o mais simples e robusto possível. Com isto em mente, o hardware resultante deve ser pequeno, leve e apresentar um pequeno consumo de energia, simplificando assim, sua aquisição, instalação e manutenção.

Para atingir os objetivos propostos o trabalho aqui apresentado seguiu um fluxo de tarefas, conforme a seguir:

1. Elaboração do software que implementa a pilha de protocolos TCP/IP.

2. Descrição do hardware de um microcontrolador ASIP com auxílio da ferramenta SASHIMI.

3. Simulação do hardware com a ferramenta MAX + Plus II e levantamento de alguns dados de desempenho.

1.1 O Estágio Atual da Pesquisa Mundial

As empresas da área de fabricação de CI's, muitas vezes fomentadas por empresas consumidoras de tecnologia, estão dedicando grandes esforços no sentido de desenvolver soluções simples e eficientes na área de comunicação de dados sobre a Internet.

Um exemplo neste sentido, é a pesquisa promovida pela Daimler-Benz Research and Technology Center dos Estados Unidos que colocou um grupo de pesquisadores no desenvolvimento de soluções de comunicação via Internet para automóveis [JAM 98]. A pesquisa visa contemplar os veículos com um produto completo e acabado, portanto muito está sendo investido no sentido de resolver problemas inerentes ao projeto. Três pontos fundamentais estão em desenvolvimento: o meio de comunicação sem fio, a arquitetura do sistema e a interface com o usuário.

Outro aspecto da pesquisa atual é o desenvolvimento de CI's que integram, cada vez mais, as funções sempre executadas por software. Um exemplo disto é o CI desenvolvido pela Osicom Technologies que integra em uma pastilha a interface Ethernet juntamente com os protocolos TCP, UDP, RARP, ICMP, HTTP, POP, SMTP e FTP chamado de *system on silicon* [VAR 98].

Algumas outras implementações da pilha de protocolos TCP/IP notáveis por sua simplicidade e economia de recursos são as desenvolvidas com os microcontroladores da Microchip [LOE 99], da Atmel [LIG 2000], e da Scenix [SCE 2000]. As três implementações colocam a pilha de protocolos em software, o que os tornam um pouco lentos e limitados, devido aos recursos escassos de memória e processamento, mas do ponto de vista de custo de fabricação, consumo de potência e de tamanho físico, são incomparáveis.

O CI mais notável sob o aspecto de conectividade com a grande rede é, sem dúvida, o S7600A da Seiko Instruments Inc. [SEI 2001], o qual possui implementado em hardware os protocolos TCP/UDP para o nível de transporte, o protocolo IP para o nível de rede e para o nível de enlace de dados utiliza o PPP. A parte física fica a cargo do projetista através de uma UART (Universal Assynchronous Receiver Transmitter) interna. Com grande parte da pilha TCP/IP implementada em hardware, a parte de aplicação fica a cargo de um pequeno microcontrolador externo, que pode ser escolhido dentre vários modelos e marcas de possível utilização com a interface disponível no CI da Seiko.

O S7600A implementa a tecnologia conhecida por *iReady*, a qual apresenta ao usuário uma extensa lista de funções (API – Application Programming Interface) para a troca de dados e gerenciamento do circuito como um todo, independente de sistema operacional, sendo fornecida em linguagem C.

Outro circuito integrado interessante nesta área é o SC12 IPC@CHIP fabricado pela empresa Beck [BEC 2001], este CI encapsula, em uma única pastilha, um processador 80186, 512 kB de memória RAM (*Random Access Memory*), 512 kB de memória Flash e um controlador Ethernet conforme o padrão físico 10Base T [SPU 2002]. Este sistema executa uma versão simplificada do sistema operacional DOS (*Disk Operating System*), e apresenta ao usuário uma ampla gama de rotinas de BIOS (*Basic Input Output System*), que além das rotinas de uso geral, apresentam funções de controle e manipulação do ambiente de rede com o conjunto de protocolos TCP/IP.

1.2 O Ambiente de Aplicação

Baseado na amplitude da aplicabilidade do protocolo de comunicação de dados TCP/IP, qualquer sistema que necessite a troca de dados ponto-a-ponto ou ponto-multiponto, que utilizar tal pilha terá, a priori, uma ampla gama de conectividade entre sistemas heterogêneos, bem como enormes facilidades de conexão física em rede. Baseado no aspecto de que embora não padronizado por RFC's (*Request for Comments)*, a camada física de ambientes que se utilizam da pilha TCP/IP possuem opções de conectividade bastante difundidas e adotadas de fato como padrões industriais. Exemplos neste sentido podem ser o padrão físico Ethernet, bem como a conexão serial assíncrona com seus protocolos de nível de enlace de dados bem definidos e mundialmente aceitos.

Com o canal de transporte de dados entre sistemas heterogêneos resolvido através da aplicação do protocolo TCP/IP e seu nível físico, resta apenas a definição da camada de aplicação, que embora bastante dependente da aplicação fim pode, de forma simplificada, ser utilizado algum protocolo já em uso no "Mundo Internet", como por exemplo o HTTP (Hiper Text Transfer Protocol).

O ambiente de aplicação proposto por este trabalho foi o de um sistema de comunicação de dados utilizado para o telecontrole e a telemedição de sistemas elétricos de potência, onde o número e a complexidade dos pontos de controle e medição possuem características apropriadas às restrições impostas pelo protocolo TCP/IP.

O sistema elétrico, de maneira geral, não possui características de tempo real, ou seja, o tempo de resposta do ambiente como um todo não é crítico, sendo observáveis intervalos de tempo entre comando e resposta na casa de segundos, tipicamente entre 3 a 10 segundos. Embora tais tempos não sejam desejáveis, são toleráveis na aplicação em que este trabalho se propõe, vindo ao encontro com a característica não determinística do protocolo TCP/IP.

Outro aspecto é a crescente demanda e geração de informações por parte dos equipamentos empregados no sistema de controle e proteção elétrico. Com o advento da digitalização dos relés de proteção, a quantidade de dados analógicos e digitais gerados por tais dispositivos vem crescendo vertiginosamente ao longo do tempo. Como exemplo, pode-se citar os relés de proteção elétrica fornecidos pela General Electric (GE) onde se encontram a disposição do usuário dezenas de milhares de informações diferentes [GEN 2001]. Baseado na quantidade de dados disponível, é de se supor que a largura de banda do canal de comunicação de dados utilizado por tais equipamentos deva ser bastante larga. Neste particular, o protocolo TCP/IP fornece características de velocidade de troca de dados baseado principalmente na camada física escolhida.

A confiabilidade dos dados transportados é outro ponto que deve ser levado em conta, porque a aplicação no tele controle de sistemas elétricos de potência pressupõe troca de dados com alta qualidade, tendo em vista que o processo é extremamente crítico sob o ponto de vista de segurança pessoal. Neste particular, estão padronizados vários esquemas de controle e correção de erro pela pilha TCP/IP em diversas camadas, redundando na troca de dados extremamente confiável.

No ambiente de aplicação de controle de sistemas de potência existem atualmente vários protocolos de comunicação de dados em utilização, com normatização internacional. Como exemplo, pode-se citar o protocolo IEC 870-101, baseado na norma de mesmo nome [INT 95], e o protocolo DNP 3.0, também baseado nesta norma internacional, embora incompatíveis entre si. Protocolos industriais estão surgindo neste ambiente, como o Modbus [MOD 96] lançado originalmente pela empresa Modicon, e o Profibus [PRO 99], além de vários outros protocolos proprietários de menor expressão.

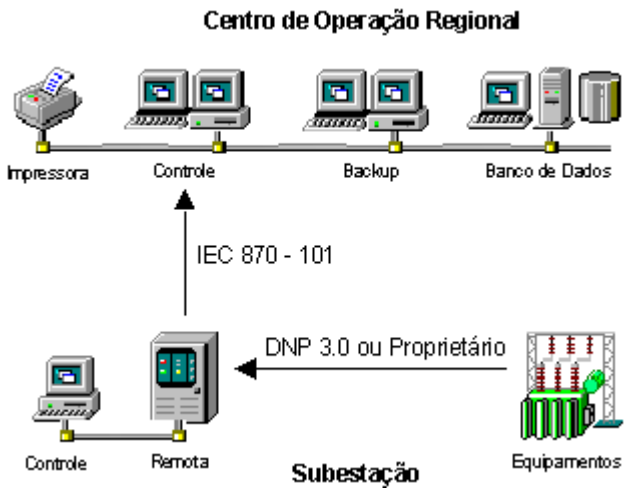

FIGURA 1.2 – Protocolos do centro de operação.

Neste ambiente com tantas possibilidades de protocolos de comunicação de dados, o papel do *gateway*, ou seja, o tradutor de protocolos, é de fundamental importância. Como ilustrado na figura 1.2, existe um protocolo de comunicação entre o equipamento de campo, tipicamente um relé de proteção elétrica digital, e a unidade terminal remota, que executa o papel de tradutor para o protocolo de comunicação utilizado pelo centro de operação. Podem ser encontrados casos onde, dentro da subestação, equipamentos de fabricantes diferentes disponibilizem protocolos diferentes para a unidade terminal remota, que deverá concentrá-los e traduzi-los adequadamente.

Existem equipamentos de proteção elétrica que estão sendo fornecidos com o protocolo de transporte TCP/IP e a camada física Ethernet [GEN 2001], mas isto ainda é uma aproximação muito recente e pouco difundida, sendo ainda muito empregados os protocolos específicos da área elétrica como o DNP e o IEC, bem como os industriais Modbus ou Profibus.

Neste ambiente, o encapsulamento de algum protocolo serial dentro da pilha TCP/IP é uma possibilidade bastante poderosa, tendo em vista a facilidade de conexão de equipamentos diversos. Um exemplo é o encapsulamento de um dos dois principais protocolos da área de controle elétrico (DNP e IEC), que utilizam no nível físico a transmissão de dados do tipo serial assíncrono com bit de início e de parada. Neste caso, o protocolo serial assíncrono é diretamente colocado na camada de aplicação da pilha TCP/IP, sendo desencapsulado do outro lado, como mostra a figura 1.3 a seguir. Considerando-se o ambiente de comunicação de dados Ethernet como serial e síncrono, então os bits de início e de parada, utilizados pelos protocolos originais, não são necessários, apresentando um ganho no desempenho.

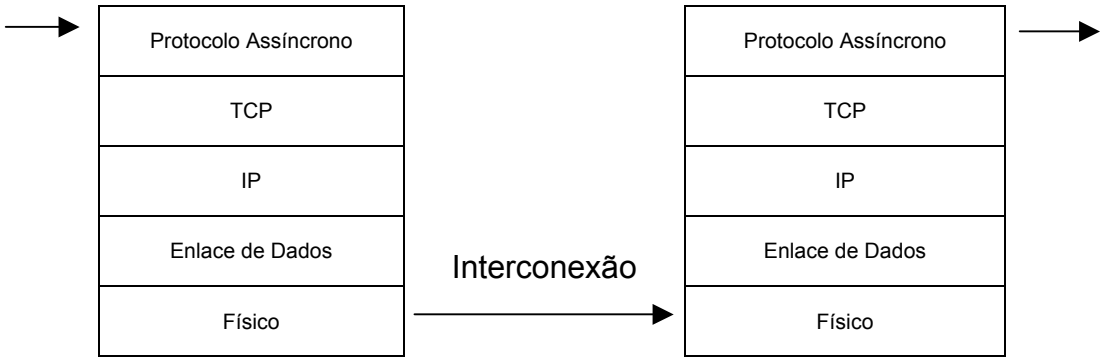

FIGURA 1.3 – Encapsulamento de protocolos assíncronos.

O encapsulamento do protocolo serial pode ser feito por um equipamento externo, ou localizar-se diretamente no equipamento de campo. Este trabalho propõe um sistema que poderá ser diretamente implementado no equipamento de campo, encapsulando o protocolo nativo ou proprietário na pilha TCP/IP, residindo no encapsulamento a grande vantagem desta abordagem, porque a partir deste momento, a conectividade, antes restrita por padrões pouco difundidos senão completamente proprietários, passam a desfrutar das vantagens da padronização e amplitude de emprego do TCP/IP.

O protocolo encapsulado na camada de aplicação poderá ser algum de utilização restrita, como tipicamente o são os protocolos utilizados na área de controle elétrico, bem como algum protocolo disponível na pilha TCP/IP, bem mais difundidos, facilitando desta forma o lado cliente do sistema.

Um exemplo neste sentido é o protocolo HTTP, que poderá ser empregado no lado cliente utilizando-se tão somente um navegador Internet como o Internet Explorer ou o Netscape Navigator.

1.3 Enfoque do Trabalho

A intenção do trabalho aqui exposto é a conexão de dois sistemas com processamento próprio, através da utilização da infra estrutura simples, econômica e acessível fornecida pelo mundo "Internet". É interessante notar que este mundo não se restringe apenas à rede física, mundialmente disponível ao usuário comum, o que por si só possui um grande valor agregado, mas principalmente seu ponto notável é a sua padronização, mundialmente aceita e utilizada.

Sob este enfoque, tal sistema de troca de informação, além da aplicação proposta por este trabalho, será útil e plenamente aplicável em pequenas redes locais, muitas vezes isoladas da grande rede mundial, onde podem ser utilizados equipamentos de interconexão de rede como modems ou hubs, com a intenção de conexão entre cliente e servidor visando a supervisão e/ou controle de pequenos processos industriais, comerciais ou até mesmo domésticos. Devido à facilidade de conexão com a rede mundial, este sistema pode ser também empregado em situações em que grandes distâncias devam ser vencidas, como no caso típico de viagens, onde o cliente se

encontra em um país e o servidor em outro. Existe também a possibilidade de emprego, conforme já abordado, em sistemas de controle industriais de grande porte, onde os equipamentos envolvidos são mais robustos e confiáveis.

O sistema que executará o papel de cliente na estrutura proposta, ou seja, o consumidor de dados, por questão de facilidade, poderá ser qualquer microcomputador que execute um sistema operacional que suporte os padrões fornecidos pela pilha de protocolos TCP/IP. No nível de aplicação será utilizado um navegador, que utilize o protocolo HTTP (Hiper Text Transfer Protocol) para apresentação dos dados.

No lado servidor, o sistema aqui elaborado fornecerá os dados requeridos pelo cliente, seguindo, de forma semelhante, os padrões estabelecidos pelas diversas RFC's, com o intuito de tornar o mais simples possível a interconexão em rede dos lados cliente e servidor. O desenvolvimento do equipamento servidor será o objetivo do trabalho aqui exposto, constituindo-se de alguns CI's (Circuitos Integrados) que, de forma compacta, simples e econômica, atenderão às requisições do cliente.

O desenvolvimento do processador TCP/IP foi feito utilizando-se uma ferramenta de desenvolvimento de processadores de aplicação específica, que toma por base o código desenvolvido em Java e produz o código VHDL (VHSIC hardware description language) correspondente ao hardware requerido. A ferramenta SASHIMI foi desenvolvida em um trabalho de mestrado da Universidade Federal do Rio Grande do Sul [ITO 2000], e fornece um processador especificamente voltado para uma aplicação descrita em Java, com um conjunto de instruções bastante particular.

Uma simplificação importante utilizada neste ambiente, visando minimizar os recursos empregados no processador TCP/IP, e em última análise reduzir a área de silício necessária durante a fase de integração em hardware, é implementado somente o lado servidor no processador. Com este artifício, grande parte do código e dos recursos necessários deixaram de ser utilizados, reduzindo o tamanho do sistema como um todo. Para tanto, parte-se do princípio que o servidor não irá solicitar uma conexão via rede, mas aceitará conexões originadas remotamente. E, após estabelecida a conexão, a troca de dados poderá se tornar bidirecional.

Para atingir tal objetivo, foram executadas as ações, conforme descrito a seguir.

- Elaboração de uma base de dados teórica, baseado na pesquisa bibliográfica sobre os temas Internet, pilha de protocolos TCP/IP, o padrão de enlace de dados Ethernet, circuitos integrados dedicados para o fim de comunicação de dados que utilizam os diversos padrões empregados na grande rede, entre outros temas correlacionados.

- Seleção e implementação de uma base de desenvolvimento de hardware constituída de uma grande rede local e alguns computadores de teste padrão PC (Personal Computer) com sistema operacional DOS, por questão de simplicidade de desenvolvimento.

- Desenvolvimento em linguagem C de uma primeira aproximação do servidor HTTP.

- Ajuste e recompilação do software desenvolvido com o fim de servir de fonte para a ferramenta SASHIMI (system as software and hardware in microcontrollers), que

gerará o código VHDL de um processador para aplicação específica, bem como o código pertinente.

- Prototipação em um FPGA (Field Programmmable Gate Array) do processador e seu software aplicativo.

O ambiente de desenvolvimento inicial foi escolhido em função da facilidade de desenvolvimento em um PC com a utilização da linguagem C. Desta forma, o hardware utilizado em rede, chamado Controlador Ethernet, pôde ser diretamente acessado, através de seus registradores, resultando em um código mais rápido e eficiente. O controlador Ethernet é um circuito integrado dedicado que se encarrega de todo o interfaceamento entre o nível de rede e o nível físico, automatizando de forma acentuada o processo de colocação dos dados no fio. Caso houvesse a opção pela Linguagem Java, fonte de entrada da ferramenta SASHIMI [ITO 2000], esta facilidade não seria verificada. Por esta razão, durante a elaboração da tradução do C para o Java, alguns artifícios foram utilizados quando da necessidade de adequação das particularidades dos dois sistemas utilizados, a saber o PC e o microcontrolador de aplicação específica.

1.4 Organização do Trabalho

Os capítulos restantes serão organizados de forma cronológica de assuntos, ou seja conforme foram sendo abordados e desenvolvidos. A ordem apresentada será como se segue:

- Capítulo 2: Este capítulo trata da codificação do protocolo com o auxílio da linguagem C. Serão apresentados aspectos relativos ao padrão utilizado nas placas de rede, com vista à implementação da camada de ligação entre o meio físico e o nível de enlace de dados. Será abordado o protocolo de comunicação que se coloca no nível de enlace de dados, e definido pela norma IEEE sob o número 802.3, bem como os principais protocolos da pilha: ARP (Address Resolution Protocol), IP (Internet Protocol)e o TCP (Transmission Control Protocol). Além disto será visto a primeira aproximação executada em C para verificação de funcionamento e viabilidade do trabalho, que foi a codificação do protocolo ICMP simplificado com o utilitário PING.

- Capítulo 3: Trata da tradução do código C para Java visando sua utilização como entrada na ferramenta SASHIMI, que gerará uma descrição em VHDL de um processador de aplicação específica, bem como o software compatível.

- Capítulo 4: Trata das conclusões retiradas do trabalho desenvolvido e apresenta algumas possibilidades de desenvolvimentos futuros.

2 O TCP/IP Implementado em C

As RFC's que padronizam os protocolos constituintes da pilha TCP/IP definem quatro camadas de serviços de rede, a saber: o nível de rede / hospedeiro, o nível de interligação de redes, o nível de transporte e o nível de aplicação. Pode ser traçado um paralelo entre o modelo TCP/IP e o modelo de referência ISO-OSI [ISO 94] que possui sete camadas, como apresentado na figura 2.1.

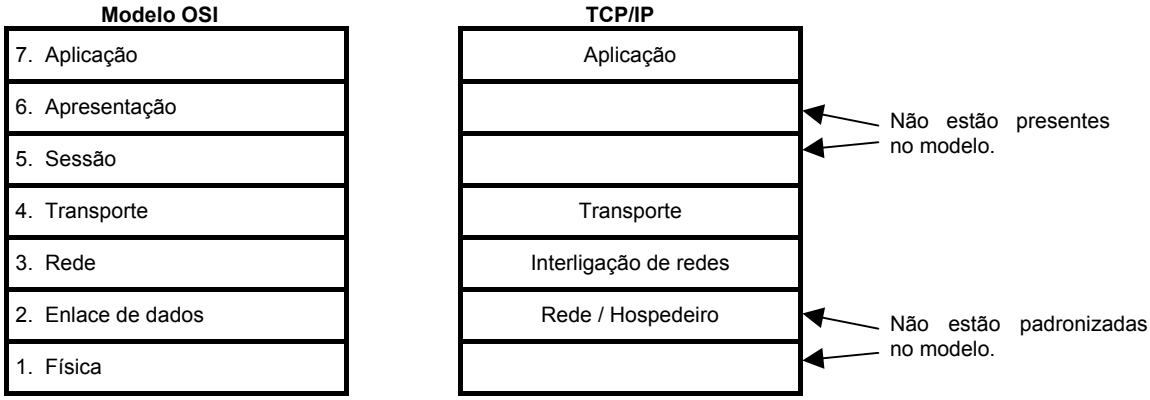

FIGURA 2.1 - Comparação entre os modelos OSI e TCP/IP.

O primeiro nível (rede / hospedeiro) não é padronizado, fornecendo uma importante característica do TCP/IP, que é a viabilidade de utilização com diferentes tipos de hardware de rede local, o que possibilita a interoperabilidade de sistemas heterogêneos, desde que seja viável a sua conexão de nível um e dois. A camada seguinte (interligação de redes) é responsável pela recepção dos pacotes que os hospedeiros injetam na rede e a sua conseqüente entrega na rede de destino, seja ela a própria rede local onde o hospedeiro gerador do pacote se conecta, ou uma rede localizada milhares de quilômetros de distância tal que, para atingi-la, o pacote deverá atravessar dezenas de redes intermediárias, configurando, neste caso, o roteamento do pacote. Essa camada pode ser comparada ao serviço de correio, onde cartas são postadas, mas nenhuma garantia de entrega ou mesmo de seqüenciamento existe, a única certeza sobre a probabilidade da entrega ou descarte do pacote roteado é que todo o esforço será empenhado pela camada, no sentido de entregar com correção os pacotes a ela confiados. O protocolo que executa os serviços deste camada é o *Internet Protocol* (IP).

A confiabilidade fica a cargo da próxima camada que fornece uma conexão fim-a-fim entre os hospedeiros fonte e destino da informação a ser trocada, independente de distância ou mesmo do número de redes intermediárias que devem ser vencidas para atingi-lo, disponibilizando mecanismos de correção de erros que, a critério do usuário, podem ou não serem utilizados. Para um fluxo de bytes fim-a-fim com conexão e correção de erros, o protocolo a ser utilizado será o *Transmission Control Protocol* (TCP) e, para o caso de não existir a necessidade de alta confiabilidade na troca de

dados, então o *User Datagram Protocol* (UDP) deverá ser utilizado por ser mais simples que o TCP. Deve-se observar que o núcleo do TCP/IP são exatamente os protocolos constituintes das camadas de nível três e quatro mais alguns outros protocolos auxiliares a estes dois níveis, além de algumas aplicações (nível sete) de implementação obrigatória segundo a padronização [MUR 98].

O primeiro passo para a implementação do sistema é a preparação e utilização do controlador Ethernet, que se encarregará de todo o protocolo de nível de enlace de dados e físico. Este controlador segue um padrão bastante difundido entre os ambientes de comunicação de dados em rede conhecido por NE2000, simplificando bastante a conexão dos protocolos de nível de rede com o cabo de rede.

2.1 O Padrão NE2000

Neste item será analisado o padrão NE2000 e o tipo de acesso a dados conhecido por E/S (entrada e saída), do ponto de vista de sua arquitetura, implementação e utilização. Sob o aspecto operacional, este tipo de acesso é muito simples, já que não necessita de hardware adicional, além do próprio circuito integrado (CI) controlador de rede. Além disto, não prevê a utilização de recursos externos, tais como memória ou outros CI's de função específica, simplificando os métodos de acesso aos dados que chegam e são enviados via rede.

O padrão NE2000 define um modo básico de acesso aos dados recebidos ou enviados pelo controlador de rede. Este modo utiliza o barramento de entrada e saída (E/S) do processador para acessar os dados que chegam e saem através da rede. Este tipo de acesso é simples e utiliza somente o hardware já disponível no CI que implementa as funções de transmissor/receptor de dados de rede, e a interface com o barramento de E/S do processador. Os controladores comerciais implementam os barramentos segundos os formatos ISA ou PCI, onde para efeitos do desenvolvimento prático do trabalho foi escolhido o formato ISA, principalmente por sua simplicidade de utilização.

Atualmente estão disponíveis CI's que implementam internamente todas as funções necessárias ao correto funcionamento em rede segundo o padrão NE2000 [NAT 95] [REA 95] [DAV 97], simplificando a implementação de um dispositivo de interface entre o hospedeiro e a rede física. Do ponto de vista da rede, ou seja, o protocolo de injeção e extração dos dados no fio, o padrão empregado é o IEEE 802.3 [IEE 96]. Do ponto de vista do driver de dispositivo o padrão é o NE2000, onde são especificados registradores de configuração, de acesso aos dados, bem como o protocolo de como extrair e enviar dados para a interface de rede.

A figura 2.2 apresenta o diagrama em blocos interno do CI controlador de rede, dando uma noção da sua arquitetura interna. As funções desempenhadas pelos diversos componentes serão objeto da análise a seguir.

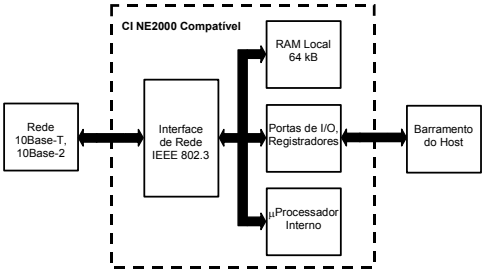

FIGURA 2.2 – Arquitetura da interface de rede.

2.1.1 Interface de Rede

A interface de rede é responsável pela adaptação de nível e codificação dos dados a serem enviados pela rede, bem como pela decodificação e filtragem dos dados recebidos desta. Além disto, este sistema executa todos os procedimentos necessários ao correto encaminhamento de pacotes, seguindo o protocolo estabelecido na norma IEEE 802.3 com o tipo de acesso ao meio físico CSMA/CD (Carrier Sense Multiple Access / Collision Detection). O preâmbulo e o CRC (Cyclic Redundant Check) do pacote Ethernet são automaticamente inseridos nos dados a serem enviados, e retirados dos pacotes que chegam conforme ilustra a figura 2.3.

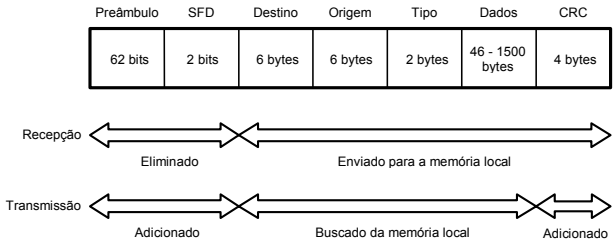

FIGURA 2.3 – Dados adicionados e removidos do quadro.

O preâmbulo é constituído de 62 bits “1” e “0” alternados, que tem a finalidade de sincronizar transmissor e receptor. O delimitador de início de quadro (SFD – start of frame delimiter) é formado por dois bits “1”, sinalizando que um novo quadro é iniciado. O CRC é calculado durante a transmissão segundo o polinômio definido na norma e adicionado ao final do quadro. Na recepção o CRC é utilizado para a verificação de integridade do quadro que chega da rede.

Esta interface mantém uma pequena memória local do tipo FIFO (First In First Out) de 16 bytes, que é utilizada para a serialização dos dados a serem enviados ou recebidos da rede. Este buffer busca ou envia os dados diretamente da memória local através do barramento interno.

2.1.2 Microprocessador Interno

O microprocessador interno ao CI é o responsável pela execução de toda a lógica envolvida no processamento dos dados e sinais presentes no barramento interno, bem como no barramento externo. Baseado nas suas rotinas, é possível executar com grande

facilidade e flexibilidade os processos envolvidos nos protocolos de rede e de driver de dispositivo, neste caso específico os padrões IEEE 802.3 e NE 2000 respectivamente.

Estão disponíveis ao usuário funções como reset, inicialização, envio de pacote pela rede, configuração dos diversos formatos de dados e sinais, entre inúmeras outras. O processador é responsável também pela organização dos dados recebidos e enviados pela interface na memória local, baseado em alguns registradores de formatação disponíveis para este fim, proporcionando desta forma, a troca de dados de forma quase transparente entre hospedeiro e rede.

2.1.3 Memória Local

A área de memória interna ao CI é onde os dados recebidos da rede são temporariamente armazenados, com o fim de serem enviados ao hospedeiro, em um segundo momento, a critério dos processos de gerência de dados que estão sendo executados no computador local. É possível que em uma rede com muito tráfego, esta área de memória estoure, quando então procedimentos especiais devem ser tomados pelo hospedeiro, com o fim de não comprometer o sistema de comunicação de dados como um todo.

Esta mesma área de memória compartilha o armazenamento dos dados enviados pelo hospedeiro à interface com o fim de serem colocados no cabo, e portanto alguns cuidados devem ser tomados quanto à definição de endereços dos dados que chegam e os que saem, visando evitar a superposição de áreas de dados. Outro recurso mantido na memória é o endereço físico da interface de rede (endereço MAC) que utiliza uma pequena área desta, com o fim de armazenar em uma E2PROM (*Electrically Erasable Programmable Read Only Memory*) o endereço de hardware e a largura do barramento utilizado pelo hospedeiro, para o caso de um barramento ISA 8 ou 16 bits.

Na figura 2.4 podemos ver os detalhes de endereçamento da memória local da interface de rede.

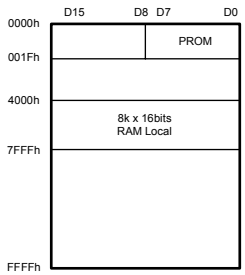

FIGURA 2.4 – Memória local da interface de rede.

Na área de endereços entre 0000h e 001Fh está localizada a E2PROM que mantém o endereço MAC do dispositivo. É interessante notar que este endereço está disponível no byte menos significativo (LSB) da palavra que forma o dado entre os endereços 0000h e 0005h (6 bytes). Também no byte LSB dos endereços entre 001Ch e 001Fh, estará armazenado um código numérico que indica o tamanho do barramento utilizado pela interface para troca de dados com o hospedeiro. Este código pode ser 42h, significando

largura de barramento de 8 bits, ou 57h, que significa barramento com 16 bits. Qualquer outro código não é aceitável pelo padrão NE2000.

A área de memória localizada entre os endereços 4000h e 7FFFh é a que será destinada como buffer dos pacotes que chegam e saem da rede através da interface. O restante da memória não é utilizado conforme o padrão estabelece.

2.1.4 Portas de E/S e Registradores

As portas de E/S e os registradores de configuração da interface de rede são diretamente mapeados em endereços de E/S do barramento do hospedeiro, e abrangem uma área de endereçamento de 32 posições. Na figura 2.5 pode-se notar a distribuição destes registradores.

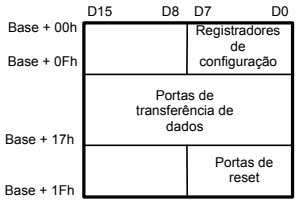

FIGURA 2.5 – Portas de E/S da interface de rede.

Os registradores de configuração da interface estão localizados entre os endereços Base + 00h e Base + 0Fh, possuindo todos 8 bits de largura, e são organizados em quatro páginas, alternáveis através do registrador de comando (CR) localizado na posição Base + 00h. O registrador de comando é o único sempre acessível nas quatro páginas. As tabelas 2.1 a 2.3 indicam a organização das três primeiras páginas de registradores, a quarta página é utilizada para testes, sendo normalmente desconsiderada.

TABELA 2.1 – Registradores de configuração, página 0.

Endereço	Leitura	Escrita
00h	CR – Command	CR - Command
01h	CLDA0 – Current Local DMA Address 0	PSTART – Page Start Register
02h	CLDA1 – Current Local DMA Address 1	PSTOP – Page Stop Register
03h	BNRY – Boundary Pointer	BNRY – Boundary Pointer
04h	TSR – Transmit Status Register	TPSR – Transmit Page Status Address
05h	NCR – Number of Collisions Register	TBCR0 – Transmit Byte Count Register 0
06h	FIFO	TBCR1 – Transmit Byte Count Register 1
07h	ISR – Interrupt Status Register	ISR – Interrupt Status Register
08h	CRDA0 – Current Remote DMA Adrress 0	RSAR0 – Remote Start Address Register 0
09h	CRDA1 – Current Remote DMA Adrress 1	RSAR1 – Remote Start Address Register 1
0Ah		RBCR0 – Remote Byte Count Register 0
0Bh		RBCR1 – Remote Byte Count Register 1
0Ch	RSR – Receive Status Register	RCR – Receive Configuration Register
0Dh	CNTR0 – Frame Alignmet Errors	TCR – Transmit Configuration Register
0Eh	CNTR1 – CRC Errors	DCR – Data Configuration Register
0Fh	CNTR2 – Missed Packets Errors	IMR – Interrupt Mask Register

TABELA 2.2 – Registradores de configuração, página 1.

Endereço	Leitura	Escrita
00h	CR – Command	CR - Command
01h	PAR0 – Physical Address Register 0	PAR0 – Physical Address Register 0
02h	PAR1 – Physical Address Register 1	PAR1 – Physical Address Register 1
03h	PAR2 – Physical Address Register 2	PAR2 – Physical Address Register 2
04h	PAR3 – Physical Address Register 3	PAR3 – Physical Address Register 3
05h	PAR4 – Physical Address Register 4	PAR4 – Physical Address Register 4
06h	PAR5 – Physical Address Register 5	PAR5 – Physical Address Register 5
07h	CURR – Current Page Register	CURR – Current Page Register
08h	MAR0 – Mutlicast Address Register 0	MAR0 – Mutlicast Address Register 0
09h	MAR1 – Mutlicast Address Register 1	MAR1 – Mutlicast Address Register 1
0Ah	MAR2 – Mutlicast Address Register 2	MAR2 – Mutlicast Address Register 2
0Bh	MAR3 – Mutlicast Address Register 3	MAR3 – Mutlicast Address Register 3
0Ch	MAR4 – Mutlicast Address Register 4	MAR4 – Mutlicast Address Register 4
0Dh	MAR5 – Mutlicast Address Register 5	MAR5 – Mutlicast Address Register 5
0Eh	MAR6 – Mutlicast Address Register 6	MAR6 – Mutlicast Address Register 6
0Fh	MAR7 – Mutlicast Address Register 7	MAR7 – Mutlicast Address Register 7

TABELA 2.3 – Registradores de configuração, página 2.

Endereço	Leitura	Escrita
00h	CR – Command	CR - Command
01h	PSTART – Page Start Register	CLDA0 – Current Local DMA Address 0
02h	PSTOP – Page Stop Register	CLDA1 – Current Local DMA Address 1
03h		
04h	TPSR – Transmit Page Status Address	
05h		
06h		
07h		
08h		
09h		
0Ah		
0Bh		
0Ch	RCR – Receive Configuration Register	
0Dh	TCR – Transmit Configuration Register	
0Eh	DCR – Data Configuration Register	
0Fh	IMR – Interrupt Mask Register	

Além dos registradores de configuração, estão acessíveis via o canal de E/S do hospedeiro entre os endereços Base + 10h e Base + 17h oito portas que podem ser utilizadas indistintamente para a transferência de dados entre o hospedeiro e a memória RAM local da interface de rede. A largura dos dados transferidos neste canal é dependente de configuração e da largura do barramento utilizado (dado disponível na E2PROM), podendo ser 8 ou 16 bits.

Na última faixa de endereços, entre Base + 18h e Base + 1Fh, é possível ativar o reset interno da interface de rede. Para atingir tal fim, basta o hospedeiro realizar uma leitura e a seguir uma escrita em qualquer dos endereços acima indicados.

Os registradores utilizados pela interface de rede tem funções distintas, como por exemplo configuração, controle de DMA (*direct memory access*) interno, controle de DMA externo, entre outras funções. Todos os registradores tem tamanho de 8 bits e são acessados via endereços de E/S do hospedeiro.

2.1.4.1 CR Command Register

Utilizado para iniciar transmissões, habilitar o DMA remoto e na troca de páginas de registradores. A tabela 2.4 demostra seu formato.

TABELA 2.4 – Configuração do registrador CR.

Bit	Símbolo	Valor	Descrição
D0	STP	0 1	Comando de reset de software
D1	STA	1 0	Comando de inicialização de software
D2	TXP	1	Comando de envio de pacote pela rede (resetado após envio)
D3	RD0	0 0 1	Comando de leitura remota (hospedeiro ← memória local)
D4	RD1	0 1 0	Comando de escrita remota (hospedeiro → memória local)
D5	RD2	0 1 1	Envio automático de pacotes (hospedeiro ← memória local)
		1 X X	Operação DMA completa ou abortada
D6	PS0	0 0	Seleciona página de registradores 0
D7	PS1	0 1	Seleciona página de registradores 1
		1 0	Seleciona página de registradores 2

2.1.4.2 ISR Interrupt Status Register

Registrador acessado para determinar a causa de uma interrupção. Pode ser mascarável através do registrador IMR (*Interrup Mask Register*). Deve ser resetado, escrevendo “1” no bit correspondente, após a ativação de uma interrupção, quando o bit cairá a “0”. A tabela 2.5 demonstra quais interrupções são monitoradas por este registrador.

TABELA 2.5 – Configuração do registrador ISR.

Bit	Símbolo	Valor	Descrição
D0	PRX	0	Pacote recebido sem erros
D1	PTX	0	Pacote transmitido sem erros
D2	RXE	0	Pacote recebido com erros (CRC, alinhamento, estouro no FIFO, pacote perdido)
D3	TXE	0	Pacote transmitido com erros (excesso de colisões, FIFO vazia)
D4	OVW	0	Estouro da capacidade do anel de recepção (CURR atingiu BNRY)
D5	CNT	0	Estouro da capacidade (MSB = 1) de um dos contadores (CNTR0, CNTR1 ou CNTR2)
D6	RDC	0	Operação de DMA remota completa
D7	RST	0	Interface entrou em estado de reset por comando ou quando o anel de recepção estoura

2.1.4.3 IMR Interrupt Mask Register

Registrador utilizado para mascarar interrupções. Deve ser escrito “1” para habilitar a interrupção correspondente, e “0” para desabilitar. Este registrador é inicializado com

todas as interrupções desabilitadas. Na tabela 2.6 podemos ver as configurações deste registrador.

TABELA 2.6 – Configuração do registrador IMR.

Bit	Símbolo	Valor	Descrição
D0	PRXE	1	Habilita interrupção de pacote recebido sem erros
D1	PTXE	1	Habilita interrupção de pacote transmitido sem erros
D2	RXEE	1	Habilita interrupção de pacote recebido com erros
D3	TXEE	1	Habilita interrupção de pacote transmitido com erros
D4	OVWE	1	Habilita interrupção de estouro da capacidade do anel de recepção
D5	CNTE	1	Habilita interrupção de estouro da capacidade de um dos contadores
D6	RDCE	1	Habilita interrupção de operação de DMA remota completa

2.1.4.4 DCR Data Configuration Register

Registrador que configura o formato de dados utilizado pela interface, conforme observado na tabela 2.7.

TABELA 2.7 – Configuração do registrador DCR.

Bit	Símbolo	Valor	Descrição
D0	WTS	0	Tamanho de palavra utilizado no DMA local e remoto igual a 8 bits
		1	Tamanho de palavra utilizado no DMA local e remoto igual a 16 bits
D1	BOS	0	MSB em D15 – D8, LSB em D7 – D0 (80x86)
		1	MSB em D7 – D0, LSB em D15 – D8 (680x0)
D2	LAS	0	DMA dual (modo de 16 bits)
		1	DMA simples (modo de 32 bits)
D3	LS	0	Modo de loop selecionado (em conjunto com D1 e D2 do TCR)
		1	Modo de operação normal.
D4	ARM	0	Comando de envio automático de pacotes desabilitado
		1	Comando de envio automático de pacotes habilitado
D5	FT0	0 0	Número de bytes presentes no FIFO para buscar/enviar dados da mem. local (2 bytes)
D6	FT1	0 1	Número de bytes presentes no FIFO para buscar/enviar dados da mem. local (4 bytes)
		1 0	Número de bytes presentes no FIFO para buscar/enviar dados da mem. local (8 bytes)
		1 1	Número de bytes presentes no FIFO para buscar/enviar dados da mem. local (12 bytes)

2.1.4.5 TCR Transmit Configuration Register

Configura a interface no que tange a transmissão de dados na rede. A tabela 2.8 ilustra as configurações deste registrador.

TABELA 2.8 – Configuração do registrador TCR.

Bit	Símbolo	Valor	Descrição
D0	CRC	0	Habilitada a geração de CRC e adição no pacote transmitido
D1	LB0	0 0	Operação normal
D2	LB1	0 1	Loop interno da interface (tipo 1)
		1 0	Loop interno da interface (tipo 2)
		1 1	Loop externo
D3	ATD	1	Permite a inibição remota da interface através da recepção de um pacote especial
D4	OFST	1	Força a mudança do algoritmo normal de backoff utilizado durante colisões

2.1.4.6 TSR Transmit Status Register

Registrador que mantém o status de uma transmissão. É limpo no início de cada nova transmissão e deve ser lido ao seu término. Na tabela 2.9 podemos observar os possíveis eventos de transmissão.

TABELA 2.9 – Configuração do registrador TSR.

Bit	Símbolo	Valor	Descrição
D0	PTX	1	Pacote transmitido com sucesso
D1	-	-	
D2	COL	1	Pelo menos uma colisão ocorreu durante a transmissão
D3	ABT	1	Transmissão cancelada por colisões excessivas (16 colisões)
D4	CRS	1	Portadora perdida durante a transmissão (transmissão não será cancelada)
D5	FU	1	Transmissão cancelada por falta de dados no FIFO
D6	CDH	1	Falha na transmissão do sinal de colisão após a transmissão de um pacote
D7	OWC	1	Colisão ocorreu após o slot time (51,2μs)

2.1.4.7 RSR Receive Configuration Register

Configura a interface no que tange a recepção de dados da rede. A tabela 2.10 ilustra as configurações deste registrador.

TABELA 2.10 – Configuração do registrador TCR.

Bit	Símbolo	Valor	Descrição
D0	SEP	1	Pacotes com erro de CRC e alinhamento são aceitos
D1	AR	1	Pacotes com menos que 64 bytes são aceitos
D2	AB	1	Pacotes com endereço de destino broadcast são aceitos
D3	AM	1	Pacotes com endereço de destino multicast são aceitos
D4	PRO	1	Habilita o modo promíscuo
D5	MON	1	Recebe os pacotes destinados à interface e com CRC correto, mas não os envia à memória (CNTR2 é incrementado)

2.1.4.8 RSR Receive Status Register

Registrador que mantém o status de uma recepção. É limpo no início de cada nova recepção e ao seu término é enviado juntamente com o pacote à memória local. Na tabela 2.11 podemos observar os possíveis eventos de recepção.

TABELA 2.11 – Configuração do registrador TSR.

Bit	Símbolo	Valor	Descrição
D0	PRX	1	Pacote recebido com sucesso
D1	CRC	0 0	Pacote recebido sem erros
D2	FAE	0 1	Pacote recebido com erro de CRC (CNTR1 é incrementado)
		1 1	Pacote recebido com erro de alinhamento e CRC (CNTR0 é incrementado)
D3	FO	1	Recepção cancelada por falta de extração de dados do FIFO
D4	MPA	1	Pacote recebido, mas por algum motivo não pode ser enviado à memória local (CNTR2 é incrementado)
D5	PHY	0	Pacote recebido por endereço físico
		1	Pacote recebido por endereço de multicast
D6	DIS	1	Receptor em modo monitor
D7	DFR	1	Sinais de detecção de portadora e de colisão gerados pela interface

2.1.4.9 Registradores de Controle do DMA Local para Transmissão

O registrador TPSR aponta para o ponto inicial do pacote na memória local, que deve ser transmitido pela interface na rede. É utilizado apenas o byte mais significativo (MSB) porque a memória local é segmentada em blocos de 256 bytes. O par de registradores TBCR0 e TBCR1 receberá o tamanho do pacote, que pode chegar a 64 kbytes, ficando por conta do programador a organização dos pacotes com sua área de dados entre 64 e 1500 bytes de tamanho, bem como a sua estruturação, segundo o formato de pacote Ethernet.

Após a inicializar dos registradores de transmissão, o bit TXP do registrador de comando (CR) pode ser setado acarretando na carga da FIFO com os dados da memória local, de forma seqüencial, com o fim de serem injetados no cabo.

2.1.4.10 Registradores de Controle do DMA Local para Recepção

A memória local é organizada em segmentos de 256 bytes com a forma de um anel, portanto mecanismos devem ser previstos para evitar a superposição de dados ainda não requisitados pelo hospedeiro com dados que chegam pela rede. Para atingir este fim quatro registradores estão disponíveis para o usuário: PSTART que delimita a posição de memória local inicial utilizada pelo anel; PSTOP que delimita o fim do anel; CURR que aponta para aposição ao longo do anel na qual deve ser escrito o próximo pacote que chega da rede; e, BNRY que aponta para o próximo pacote que deve ser retirado do anel pelo hospedeiro utilizando o canal de DMA remoto.

Ao receber um pacote a interface verifica a posição atual de escrita (CURR) e a compara com PSTOP. Caso seja igual, então foi atingida a posição máxima permitida ao anel e CURR será igualado a PSTART, ou seja, localizado no início da área de memória novamente. Após é comparado CURR com o conteúdo de BNRY, que se

forem iguais indicará que acontecerá sobrescrita em dados ainda não requisitados pelo hospedeiro. Neste caso a recepção é suspensa e uma interrupção gerada, com o fim de avisar o hospedeiro que os dados devem ser retirados do anel, utilizado-se para isto um processo de reinicialização da interface definido pelo fabricante.

2.1.4.11 Registradores de Controle de DMA Remoto

O DMA remoto é utilizado com o fim de extrair dados da memória interna da interface de rede, e enviá-los à memória do hospedeiro e vice-versa. Os registradores de controle desta transferência são RSAR0 e RSAR1 que apontam para a área da memória local onde deve se iniciar a transferência e, RBCR0 e RBCR1 que contêm o tamanho da área de dados a ser transmitida.

Após preencher convenientemente o valor destes registradores, poderá ser emitido um comando de leitura remota, de escrita remota ou de envio de pacote, utilizando-se para tanto os bits RD0, RD1 e RD2 do registrador de comando (CR).

Quando for emitido um comando de leitura remota cada posição de memória local será escrita na porta de E/S da interface aguardando ser lida pelo hospedeiro via um comando de leitura de E/S do processador. A partir do ponto inicial indicado por RSAR, a leitura continuará seqüencialmente até atingir a quantidade máxima de dados indicada por RBCR. O comando de escrita remota tem um procedimento semelhante, mas com a direção do fluxo de dados inversa, ou seja, o hospedeiro escreve os dados a serem enviados para a memória local na porta de E/S, e estes dados são encaminhados, via o canal de DMA remoto, para a memória local.

O comando de envio de pacote possibilita a automação do processo de transferência de dados entre a memória local e o hospedeiro, movimentando adequadamente os ponteiros RSAR, RBCR e BNRY com o fim de enviar o próximo pacote que se encontra na memória local para o hospedeiro, liberando o programador da codificação de rotinas de gerência deste tipo de transferência. Após o ajuste deste bit do registrador de comando (CR), o processador interno da interface de rede executa todos os procedimentos necessários visando a transferência de um pacote completo da memória local para a memória do hospedeiro. Fica a cargo do programador a codificação de uma rotina de leitura da porta de E/S da interface, com o fim de colocar os dados disponibilizados, de forma seqüencial, na memória do hospedeiro.

2.2 O Código Pingador

A primeira aproximação com a pilha de protocolos TCP/IP foi executada utilizando a seqüência de testes prevista no utilitário PING (*Packet Internet Groper*). Para tanto, foi implementado com a utilização da linguagem C [KER 86], [SCH 97], o código que responde a uma solicitação de eco. Desta forma, o sistema reproduz o elo secundário previsto no teste de funcionalidade da pilha TCP/IP.

A implementação foi executada utilizando um PC com processador 80386 e 16 MB de memória RAM. Uma placa de rede que emula o padrão NE2000 foi peça fundamental, uma vez que desta forma tornou-se viável o acesso direto ao hardware de rede,

utilizando o padrão, evitando desta forma, o uso de drivers complexos ou outros artifícios junto ao sistema operacional do microcomputador (DOS 6.22).

A linguagem C foi escolhida devido à sua facilidade de acesso direto ao hardware, utilizado no dispositivo de conexão de rede (Controlador Ethernet). Com o conjunto PC, DOS e C, o ambiente de desenvolvimento tornou-se bastante flexível e relativamente fácil de programar. Esta facilidade de acesso ao hardware não estaria disponível, durante a fase de desenvolvimento, caso se optasse pelo uso da linguagem Java, o que fatalmente levaria à utilização de drivers de dispositivo, transformando-se em um empecilho a mais no código resultante.

Como diretriz básica do trabalho ficou estabelecido que o código, bem como os recursos de hardware utilizados, deveriam ser os mais simples possível, evitando portanto, sofisticações desnecessárias ao correto funcionamento do protocolo. Esta simplificação é baseada no fato de que o código resultante, devidamente traduzido para Java, servirá como fonte de entrada para a ferramenta SASHIMI, que por sua vez produzirá o código necessário para a sintetização em hardware do processador de aplicação específica TCP/IP. Tendo em vista as limitações que o hardware impõe quanto à área de silício ocupada pelo código produzido pela ferramenta utilizada, ficou estabelecido que quanto menos recursos fossem utilizados na fase de desenvolvimento, mais simples e exeqüível seria a integração em hardware dos resultados finais. Para tanto, nesta primeira implementação um pequeno subconjunto do universo de possibilidades dos protocolos da pilha envolvidos no processo de PING foram utilizados.

A própria linguagem de programação foi utilizada tendo em mente a simplicidade do código resultante, procurando-se, na medida do possível, evitar o uso de construções complexas. Funções de muitas bibliotecas disponíveis no C foram evitadas, bem como a utilização de operadores mais sofisticados, como por exemplo, a multiplicação, a divisão, entre outros.

2.2.1 O Utilitário PING

Visando a depuração e localização de defeitos na rede TCP/IP [COM 98], foi padronizada uma ferramenta de depuração baseada na solicitação de eco (*echo request*) e a resposta ao eco (*echo reply*). Um hospedeiro envia uma solicitação de eco, e o hospedeiro destino deve responder a esta solicitação retornado à origem um pacote com determinada formatação.

A mensagem que flui pela rede como resultado de uma requisição possui uma área de dados opcional, que caso exista no pacote gerado na origem, deve retornar exatamente igual no pacote de resposta.

Para o correto encaminhamento da solicitação PING todos os protocolos inferiores ao aplicativo devem estar habilitados e funcionais, sendo portanto um teste efetivo quanto à funcionalidade, não só dos hospedeiros origem e destino, mas também de toda a interligação de redes que eventualmente exista entre eles.

O utilitário PING se utiliza do protocolo ICMP (*Internet Control Message Protocol*) [POS 81a] para executar o teste de funcionalidade em rede. O ICMP prevê uma solicitação de eco e uma resposta a esta solicitação, utilizando um pacote como o mostrado na figura 2.6.

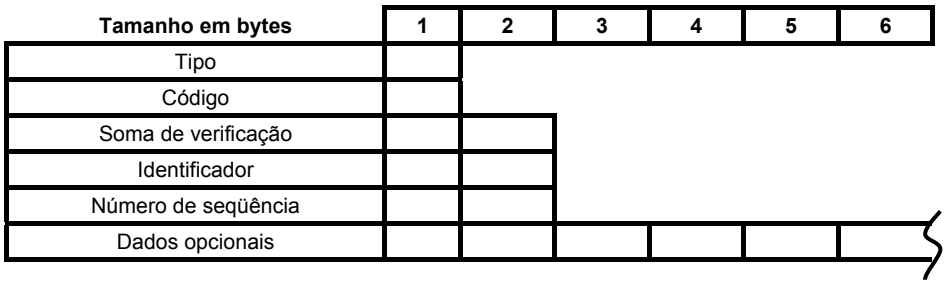

FIGURA 2.6 - Formato da mensagem de eco (ICMP).

O campo *Tipo* possui o valor 8 para a solicitação de eco, ou o valor 0 para a resposta. O campo *Código* sempre tem o valor 0, a *Soma de verificação* é gerada na origem do pacote e utilizada pelo destino para a verificação de erros de transmissão. Os campos *Identificador* e *Número de seqüência* são utilizados pelo hospedeiro que solicita o eco com o fim de combinar suas solicitações com as respostas adequadas.

Além do protocolo ICMP o PING verifica a funcionalidade do protocolo IP (*Internet Protocol*) [POS 81], sobre o qual o ICMP é transportado, e no caso do nível físico ser o Ethernet, então o protocolo ARP (*Address Resolution Protocol*) [PLU 82] deverá estar operacional também.

2.2.2 O Hardware de Rede

A implementação foi baseada no padrão NE2000, que possibilita o acesso ao hardware de rede diretamente pelo software elaborado em C durante a fase de desenvolvimento, dispensando a utilização de drivers de dispositivo de qualquer espécie. Vários fabricantes de Circuitos Integrados (CI) produzem controladores de rede que implementam o padrão na interface entre o software e os registradores de nível físico, disponibilizando seus manuais, com toda a informação necessária ao correto acesso de seus controladores.

Para o bom funcionamento do hardware de rede, alguns registradores devem ser inicializados com valores específicos, dependendo da forma de acesso que o hospedeiro utilizará no tocante à transferência de dados da rede e a forma de acesso à memória interna (memória local) do controlador. O modo de operação padrão do NE2000 é o formato de acesso aos dados via E/S, sendo este o formato padrão de alguns controladores comerciais que seguem o padrão. Neste formato toda a troca de dados entre a memória do hospedeiro e a memória local do controlador Ethernet, será baseada em comandos de leitura ou escrita em E/S do processador utilizado no hospedeiro.

A partir deste ponto a programação dos diversos registradores do controlador seguem as especificações NE2000 para o correto funcionamento da troca de dados entre hospedeiro e rede. A rotina de preparação do hardware de rede utilizado no código envia valores adequados aos registradores da placa de rede visando colocá-la operacional na rede local Ethernet.

Outro aspecto importante da rotina de inicialização é a preparação do hardware de tratamento de interrupções do microcomputador, visando iniciar o tratamento das interrupções geradas pela placa de rede. Com esta intenção é ajustado o vetor de interrupções do sistema operacional com o endereço da nova rotina de tratamento de

interrupções (ISR – *Interrupt Service Routine*), que substituirá a funcionalidade original da rotina de tratamento de interrupções do sistema operacional, por uma nova funcionalidade, de acordo com as necessidades do controlador de rede utilizado.

Esta ISR prevê o tratamento de três eventos básicos, conforme demostra o fluxograma apresentado na figura 2.7:

- A recepção de um pacote da rede sem erros;
- A transmissão para a rede de um pacote sem erros;
- A ocorrência de um erro de transmissão.

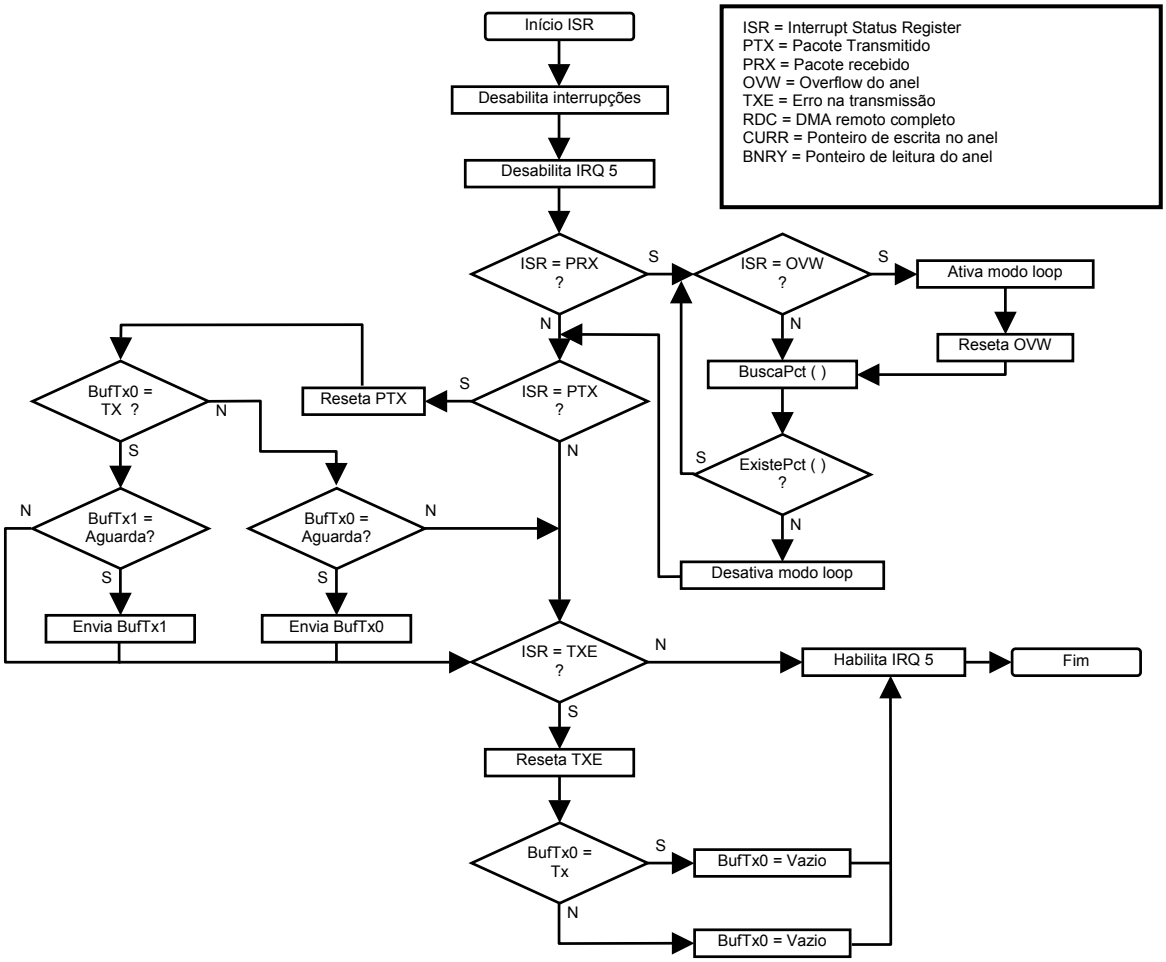

FIGURA 2.7 – Fluxo da rotina de tratamento de interrupções.

2.2.3 Recepção de Dados da Rede

O evento de recepção de um ou mais pacotes da rede é tratado através da retirada dos dados da memória local do controlador de rede, onde os pacotes são automaticamente colocados durante sua recepção, e aguardam serem retirados pelo hospedeiro através de operações de E/S do processador. Eventualmente, pode ocorrer que a memória interna estoure devido à chegada de uma quantidade maior de dados da rede que o hospedeiro pode retirar da memória interna. Neste caso, uma interrupção é gerada pelo controlador, e como conseqüência, este é colocado em modo de laço, evitando o recebimento de mais dados. Com este artifício é gerada a condição para que o hospedeiro esvazie a memória local. A seguir o processo de recepção de dados é reinicializado, evitando desta forma, a sobrescrita dos dados já recebidos da rede.

As funções diretamente envolvidas neste processo são `BuscaPct()` e `ExistePct()`. A primeira função executa as operações necessárias de E/S visando retirar da memória local do controlador de rede o pacote recém chegado, e transferi-lo à memória principal do microcomputador. Após este processo, é feita uma análise do conteúdo do pacote visando determinar se seu destino é a máquina local. Após, é feito o encaminhamento à rotina de tratamento de protocolo adequada. No caso do software pingador, apenas duas possibilidades de protocolo estão disponíveis: ARP e ICMP. A figura 2.8 ilustra o processo da rotina `BuscaPct()`.

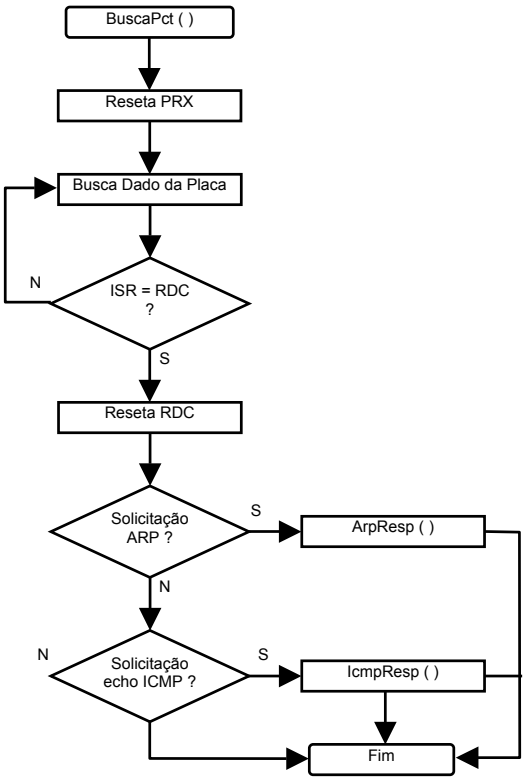

FIGURA 2.8 – Fluxo da rotina BuscaPct().

A função `ExistePct()` executa o procedimento de verificação se a memória local do controlador de rede está vazia. É utilizada durante o processo de retirada de dados da memória local evitando, desta forma, que pacotes sejam deixados no controlador,

ocupando espaço de memória, e deixando de ser tratados pelo protocolo. A figura 2.9 ilustra seu processo interno.

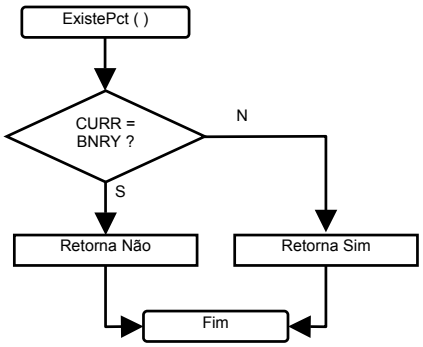

FIGURA 2.9 – Fluxo da rotina ExistePct().

2.2.4 Envio de Dados para a Rede

A função responsável pela transmissão de dados para a rede é a `EnviaPct()`. Na memória local do controlador de rede existem duas áreas onde devem ser colocados os pacotes que serão enviados. Após a carga dos dados na memória local do controlador de rede, através de operações de E/S, este é ajustado para o envio destes dados através da rede.

As duas áreas de memória para a transmissão podem ser utilizadas de forma entrelaçada, de maneira que enquanto uma está transmitindo seus dados através do controlador, a outra pode ser carregada com novos dados vindos do hospedeiro. Desta forma pode ser obtido um substancial aumento na performance de transmissão de dados. O processo desta função está ilustrado na figura 2.10.

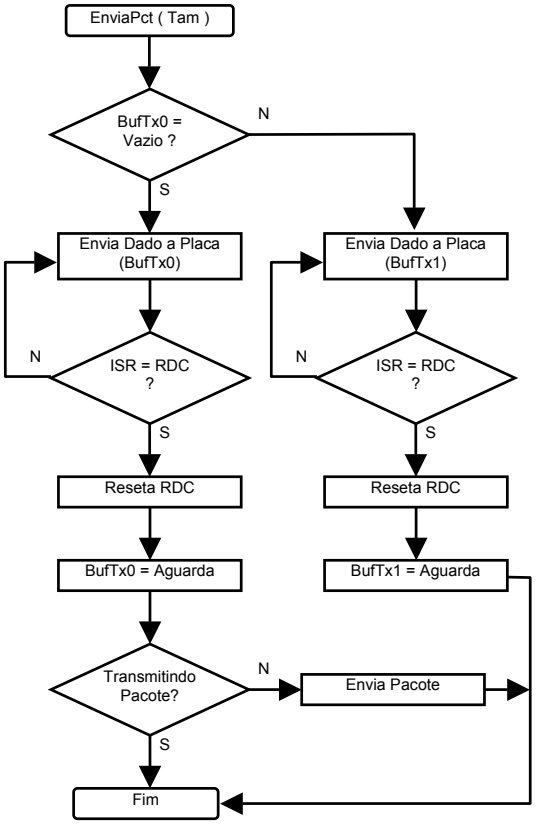

FIGURA 2.10 – Fluxo da rotina EnviaPct().

O primeiro passo é verificar qual a área de memória está vazia para o envio de um novo pacote. Após, a operação de E/S é executada transferindo os dados da memória do hospedeiro para a memória local do controlador. A seguir, se possível, o controlador de rede é imediatamente ajustado para o envio do pacote pela rede. Caso o envio imediato não seja viável, porque o canal de transmissão de dados via rede está ocupado, então a rotina de tratamento de interrupções se responsabilizará pelo seu envio no devido tempo.

2.2.5 O Protocolo

A implementação do software pingador é bastante simplificada no que tange os recursos dos protocolos aplicados, sendo desenvolvido apenas o necessário para o correto funcionamento com as características que o utilitário PING deseja encontrar.

Sob este ponto de vista apenas três funções se tornaram necessárias, a saber:

- A função de prova da soma de verificação dos cabeçalhos, `CheckSum()`;
- A função de elaboração da resposta a uma solicitação de resolução de endereço físico (ARP), `ArpResp()`;
- A função de elaboração da resposta a uma solicitação de eco (ICMP), `IcmpResp()`.

2.2.6 Soma de Verificação

Segundo o protocolo IP, o cabeçalho de cada pacote possui um campo de verificação da integridade física dos dados ali disponíveis. Com este artifício pode ser detectado um pacote com dados corrompidos na área de cabeçalho, o que fatalmente acarretará no seu descarte. Embora o nível de enlace de dados adotado (Ethernet) providencie uma forma de verificação de integridade do pacote como um todo, através de cálculo de CRC (*Cyclic Redundant Check*), que por si só é altamente eficiente na detecção de erros de transmissão, o IP prevê mais este nível de segurança quanto aos dados que chegam. O controlador de rede fornece de forma automática a verificação dos pacotes Ethernet que chegam, bem como o cálculo do CRC para os pacotes que saem.

O tipo de teste de verificação adotado, para o caso do IP, é a soma em complemento de um, aplicada somente sobre a área de dados do cabeçalho IP nos pacotes que chegam. O procedimento adotado a partir da detecção de um pacote defeituoso é o seu descarte. O mesmo processo é adotado, mas de forma inversa, sobre os pacotes que saem. Neste caso a soma de verificação é calculada e adicionada ao cabeçalho IP no campo adequado.

2.2.7 A Resposta ARP

Na rede local, a única forma de localizar um determinado hospedeiro é através do protocolo de resolução de endereço IP [POS 81] em endereço físico, viabilizando assim a remessa de um pacote de dados no nível de enlace de dados de um hospedeiro a outro. Para maiores detalhes do protocolo ver o desenvolvimento ARP específico, na seção 2.5. Com este fim foi especificado o protocolo ARP [PLU 82], que se tornou parte integrante da pilha TCP/IP nas redes locais que seguem a especificação Ethernet.

Este protocolo foi implementado no software pingador, visando sua correta funcionalidade na rede local. A função `RespArp()` recebe o pacote com uma solicitação de resolução de endereço remetido para a máquina local, e constrói uma resposta adequada. A seguir envia esta resposta à função de envio de pacotes para a rede, com o fim de que a resposta chegue ao hospedeiro originador da requisição.

De posse do dado necessário (endereço físico) para o envio de um pacote Ethernet, o hospedeiro de origem pode então executar a entrega do pacote de solicitação de eco, que em última análise é o objetivo do software implementado. O ARP implementado possui apenas uma fração da funcionalidade prevista no padrão, visando exclusivamente a economia de recursos de hardware e software.

2.2.8 A resposta ICMP

Ao receber um pacote IP com uma solicitação de eco, a função `BuscaPct()` verifica se o destino é o endereço IP local, se a soma de verificação do cabeçalho IP está correta, se a versão do protocolo IP é a versão corrente e, por fim, se a soma de verificação do cabeçalho ICMP está certa. A soma de verificação do cabeçalho ICMP segue as mesmas regras do cabeçalho IP quanto à abrangência e formato.

Após esta verificação o pacote é enviado para a função de elaboração de resposta de eco, que montará a resposta conforme o protocolo ICMP prevê. Basicamente os

endereços de origem e destino são invertidos no pacote e as somas de verificação dos cabeçalhos IP e ICMP são recalculadas. A seguir a função de envio de pacotes é chamada, injetando a resposta elaborada localmente na rede, e o processo finaliza.

2.3 Funções Auxiliares

Durante o progresso da implementação do software de emulação do protocolo TCP/IP no PC, foi surgindo a necessidade de codificação de algumas funções acessórias, mas de fundamental importância para o correto desempenho do sistema como um todo. Tais funções serão objeto de análise desta seção, visando a compreensão correta da forma de implementação do sistema.

As funções elaboradas são as a seguir relacionadas:

- Função de cálculo de soma de verificação
- Função de inversão de bytes de uma word
- Função de temporização
- Função de soma de 32 bits

Estas funções tornaram-se necessárias em sua maioria devido às limitações impostas pelo sistema operacional utilizado (DOS 6.22). Por outro lado a futura implementação em hardware (VHDL) do código proposto pode ser vantajosa sob o aspecto de que não são utilizadas rotinas proprietárias do sistema operacional utilizado para o desenvolvimento, as quais não estariam presentes na implementação do hardware final.

2.3.1 O Cálculo de Soma de Verificação

A forma de cálculo de soma de verificação utilizada pelos protocolos IP (*Internet Protocol*), ICMP (*Internet Control Message Protocol*) e TCP (*Transmission Control Protocol*) [COM 98] [COM99] é a mesma segundo as respectivas RFC's (*Request For Comment*) que os padronizam [POS 81] [POS 81a] [POS 81b]. Portanto, surgiu de imediato a necessidade de desenvolvimento de uma função acessível aos três protocolos que gerasse tal soma de verificação.

O formato desta verificação é baseado em palavras de 16 bits que são separadas em seus bytes menos significativos (LSB) e nos mais significativos (MSB). A seguir são somados todos os LSBs da área de dados que se deseja fazer a verificação. De forma semelhante os MSBs também são somados entre si.

Ao final desta operação restam dois números que representam a soma dos bytes LSB e a soma dos bytes MSB, mas estes números devem possuir uma largura de 8 bits, portanto a seguir o dígito de "vai um", que eventualmente tenha surgido na soma LSB, deve ser acrescido ao MSB. De forma semelhante o "vai um" do MSB deve ser acrescido ao LSB, restando no fim dos cálculos duas quantidades do tamanho de um byte cada.

A seguir as duas quantidades devem ser agrupadas novamente, voltando a possuir 16 bits de largura, e a quantidade final deve ser invertida bit a bit, resultando finalmente na soma de verificação da área de dados enviada para a função de cálculo.

A soma de verificação possui uma característica interessante para o protocolo que é a de verificação de integridade dos dados recebidos da rede. Baseado nesta característica, ao ser executada tal verificação sobre os dados o resultado da operação deverá ser zero, desde que se inclua na área de dados a soma de verificação que foi recebida juntamente com o pacote de dados. A figura 2.11 ilustra o fluxo desta rotina.

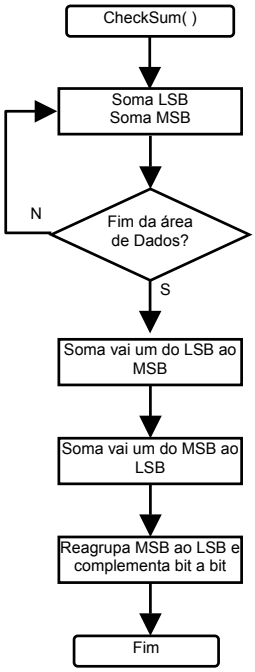

FIGURA 2.11 – Fluxo da rotina de cálculo de soma de verificação.

2.3.2 Função de Inversão MSB – LSB

A ordem de armazenamento dos bytes presentes na memória do PC é conhecida como *Little Endian*, isto é, na posição de memória com endereço de menor valor armazena o byte de maior valor de uma palavra, e na posição de memória de maior valor encontra-se o byte de menor valor. Por convenção os dados que fluem na rede, que utilizam a pilha TCP/IP, seguem o formato conhecido como *Big Endian*, exatamente o oposto encontrado no PC.

Por conseqüência, sempre que for necessária a troca de dados de 16 bits entre o hospedeiro e a rede, uma conversão de formato deverá ser utilizada, invertendo o byte MSB com o LSB. Desta forma teremos no hospedeiro o formato LSB – MSB, ao passo que na rede o formato será MSB – LSB.

Para resolver este problema, uma pequena função de conversão foi implementada, chamada de `Inverte()`, que tem o objetivo de retornar a palavra que recebeu como argumento, invertida nos seus bytes MSB e LSB.

Com a utilização deste artifício fica simples o processo de, por exemplo, realizar a soma de dois números inteiros de 16 bits de largura, recebidos da rede pelo processador do PC, e a seguir retornar o resultado de 16 bits novamente para a rede. É importante notar que esta preocupação não atinge valores codificados em 8 bits, onde neste caso a aritmética com dados vindos da rede é direta.

2.3.3 Função de Temporização

Temporização é um requisito fundamental no protocolo TCP/IP, uma vez que vários aspectos se baseiam no tempo decorrido entre dois eventos. Por exemplo, em diversas situações pode-se deparar com tempo de vida de uma determinada informação. Mais especificamente, os dados contidos na tabela ARP possuem uma determinada vida útil, após a qual devem ser descartados. O protocolo TCP utiliza muito temporizadores com o fim de determinar tempos de espera por alguma informação que esteja para chegar pela rede.

Com o fim de prover uma fonte de tempo ao programa desenvolvido, e não depender do sistema operacional para isto, foi implementado um temporizador que cadencia todo o sistema, sendo utilizado apenas o temporizador do computador hospedeiro como fonte da constante de tempo. Mais tarde, quando da implementação em hardware, um temporizador simples poderá facilmente ser utilizado em substituição ao do computador hospedeiro sem grandes alterações na lógica proposta.

O temporizador do PC fornece a constante de tempo com uma cadência de 18,21Hz, ou mais precisamente, uma interrupção do processador (IRQ 0) a cada 54,9ms [SHA 95], [PRE 99]. O vetor de interrupções foi ajustado para que, juntamente com as funções normais do sistema operacional quando da ocorrência da interrupção 0, fosse processado um pequeno fragmento de código pela ISR (*Interrupt Service Routine*).

O código executado inicialmente desabilita todas as interrupções, e a seguir incrementa algumas variáveis do programa que terão seus valores ajustados de acordo com o tempo decorrido. Uma variável para a constante de tempo do timer do PC, atualizada a cada 54,9ms, uma variável para os segundos, uma para os minutos e uma para as horas.

Após atualizar convenientemente as variáveis contadoras de tempo, as interrupções do PC são restauradas e o controlador de interrupções do PC (PIC – *Programmable Interrupt Controller*) é avisado do fim do tratamento desta interrupção.

A partir da atualização constante destas variáveis de tempo, os diversos protocolos e funções que necessitem de medição de tempo decorrido, podem manter variáveis auxiliares, e através de comparação, incremento ou decremento, baseados nas variáveis principais de contagem de tempo, terão uma precisão razoável na noção de tempo transcorrido entre dois eventos quaisquer.

A lógica da função de atualização das variáveis temporizadoras é intencionalmente simples e curta, tendo em vista que esta função será executada aproximadamente 18 vezes por segundo, e todas as interrupções do PC estarão inibidas durante sua execução. A figura 2.12 a seguir dá uma idéia da forma de implementação da função de temporização.

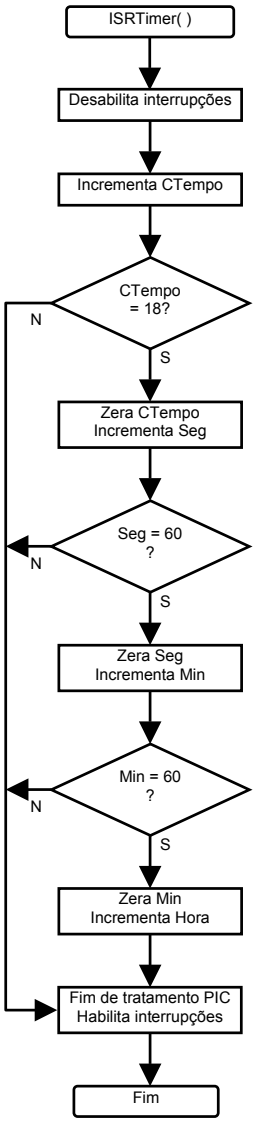

FIGURA 2.12 – Fluxo da rotina de temporização.

2.3.4 Função de Soma de 32 Bits

O sistema operacional utilizado para a implementação do sistema TCP/IP foi o DOS 6.22, que é um sistema operacional de 16 bits. Este sistema operacional não executa funções que envolvam inteiros de 32 bits de tamanho.

Algumas operações previstas no protocolo TCP utilizam dados de 32 bits, e para executar as rotinas pertinentes adequadamente, uma função de soma de 32 bits foi codificada e recebeu o nome de `Soma()`. Esta função recebe como argumento as palavras mais significativa e menos significativa do inteiro de 32 bits correspondente ao primeiro operando da soma. Recebe também um valor de tamanho de 16 bits que será o segundo operando da operação. A seguir o segundo operando é adicionado à palavra menos significativa da palavra de 32 bits (primeiro operando), e verificado se a operação produziu o bit de "vai um". No caso afirmativo, então a palavra mais significativa do primeiro operando deve ser adicionada de um, caso negativo, nada deve ser feito, e o resultado da soma estará pronto.

Os campos do cabeçalho TCP que utilizam inteiros de 32 bits são os dos números de seqüência e de reconhecimento, que para o caso aqui apresentado, nunca receberão ou enviarão um segmento TCP com mais que 1460 bytes de dados. Isto ocorre devido à área de dados máxima do quadro Ethernet ser de 1500 bytes, daí subtraindo-se o cabeçalho IP (20 bytes) e o cabeçalho TCP (20 bytes) obtém-se 1460 bytes. Adicione-se a isto que os quadros de aplicação transmitidos neste trabalho são bastante sucintos, e nunca atingirão os 1460 bytes disponíveis .Por este motivo a função de soma de 32 bits que será utilizada para gerar os números de reconhecimento e de seqüência TCP, tem como segundo operando um inteiro de 16 bits. Isto ocorre porque não são necessários mais que 11 bits para representar 1460, o maior número que poderá ser acrescido a estes campos.

Os protocolos utilizados na pilha TCP/IP, implementados neste trabalho, serão objeto de análise das seções seguintes, apresentando suas aproximações e simplificações visando atingir o menor tamanho de código possível.

2.4 O Padrão Ethernet

O padrão Ethernet é possivelmente o protocolo de nível dois mais difundido e utilizado, sendo normalizado pelo IEEE sob o número 802.3 [IEE 96]. Possui o endereço de hardware de seis bytes codificado diretamente sobre a interface de rede, com numeração exclusiva e nunca repetida. O formato do quadro Ethernet é apresentado na figura 2.13.

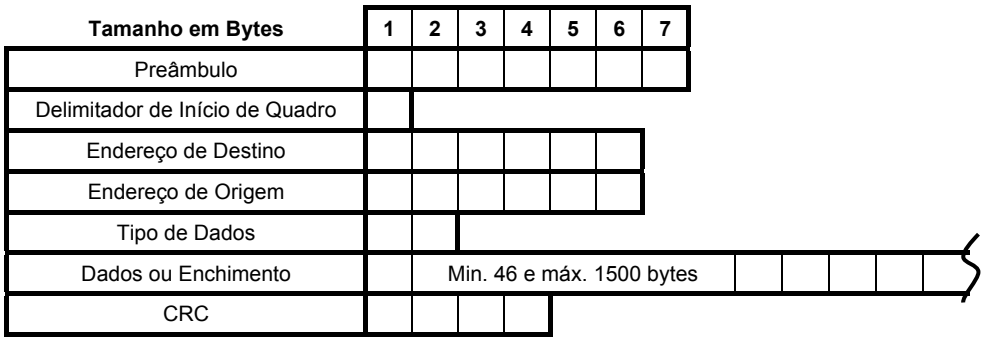

FIGURA 2.13 - Cabeçalho do quadro Ethernet

O *Preâmbulo* é constituído de sete bytes com o padrão '10101010' com o objetivo de sincronizar o receptor e o *Delimitador de Início de Quadro* possui o padrão '10101011'. O *Endereço de Destino* vem antes do *Endereço de Origem*, com a finalidade de um rápido entendimento por parte do hardware receptor, se o quadro se destina a ele e, portanto, deve ser completamente recebido e processado, ou deve ser imediatamente ignorado. Os dois bytes seguintes definem o tipo de protocolo que está encapsulado no quadro, com a finalidade de um correto encaminhamento para a rotina de software pertinente, abrindo desta forma a possibilidade de que numa mesma rede trafeguem protocolos completamente diferentes sem possibilidade de interferência mútua.

A seguir encontram-se os dados que o quadro carrega, devendo ser observado os limites máximo (MTU) e mínimo de um quadro Ethernet. Caso os dados encapsulados possuam menos que 46 bytes, então um enchimento deve ser utilizado para atingir o mínimo exigido. A seguir existem 4 bytes do código de verificação de erros que é do tipo CRC.

2.4.1 Ethernet no CI Controlador de Rede

O Circuito Integrado controlador de rede utilizado encapsula em hardware praticamente todas as funções relativas ao nível físico e de enlace de dados, tornado desta forma o processo de envio e recepção de um pacote pela rede bastante simplificado.

Tais funções são completamente transparentes ao usuário, mas devem ser lembradas principalmente devido à sua complexidade. Caso devessem ser desenvolvidas pelo projetista que implementará o uso do controlador de rede, certamente aumentariam em muito a complexidade do código resultante, de forma a talvez inviabilizar o projeto, principalmente em se considerando os escassos recursos de memória e processamento, normalmente presentes nas implementações mais econômicas.

As funções integradas no CI controlador são descritas sucintamente a seguir:

- **Adaptação do sinal de entrada**: consiste de vários filtros e circuitos reconhecedores do sinal típico presente no cabo de rede. Estes adaptadores garantem que sinais resultantes de interferências não sobrecarreguem os circuitos seguintes.

- **Detetor de colisões**: Reconhece e trata as colisões ocorridas no cabo de rede, tanto do ponto de vista físico, ou seja, o tratamento de *time out* e retransmissão, quanto do ponto de vista lógico, com aviso ao nível superior de uma transmissão bem sucedida ou abortada por excesso de colisões.

- **Detetor de link**: Gera o sinal de reconhecimento do estado de uma conexão de rede, bem como reconhece e sinaliza o estado atual da conexão de rede, baseado no sinal gerado pelo equipamento remoto.

- **Codificador / decodificador Manchester**: Executa a codificação do sinal digital a ser transmitido no cabo de rede, segundo o código Manchester, e na recepção de um sinal da rede executa a sua decodificação.

- **Gerador / verificador de CRC**: Módulo que calcula o CRC de um pacote que deve ser enviado pela rede, adicionando-o no local correto. No retorno de um pacote da rede executa a verificação do CRC, descartando dados defeituosos.

- **Reconhecimento de endereço físico**: O reconhecimento do endereço de destino de um pacote que chega é automaticamente executado por este módulo, com o descarte de pacotes que não sejam destinados ao hospedeiro local.

- **Montagem do pacote**: O pacote de dados a ser enviado, previamente colocado na memória do controlador, é montado de forma a poder ser enviado com o seu preâmbulo e CRC adicionados. Na recepção o preâmbulo é descartado e o CRC verificado.

Fica a cargo do projetista a montagem do pacote de dados a ser enviado, incluindo os endereços MAC de destino e origem, pertencentes ao cabeçalho Ethernet, bem como o código do protocolo carregado no quadro. A seguir através de operações de DMA próprias do controlador os dados devem ser armazenados na memória interna para processamento e envio pelo controlador de rede. As operações de envio para a memória

local do controlador de um pacote que deve ser enviado pela rede, e a retirada de um pacote que chega foram abordadas em detalhes anteriormente neste capítulo.

2.4.2 A Implementação do Protocolo

Baseado no exposto no item anterior, notamos que pouco fica a ser feito no processamento dos dados do quadro Ethernet, devido à grande automação que o controlador de rede utilizado proporciona.

Como resultado a função que prepara o cabeçalho Ethernet é bastante simples, como podemos observar na figura 2.14 a seguir.

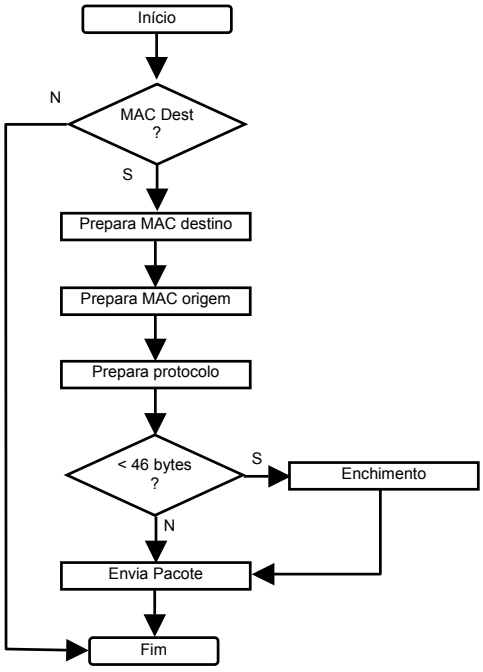

FIGURA 2.14 – Fluxo do protocolo Ethernet

Inicialmente é verificado se o endereço físico de destino foi validado pelo protocolo ARP, através de seu cache. Caso negativo o pacote não poderá ser enviado devido à falta do endereço do hospedeiro de destino. Caso o endereço esteja validado, então o restante do cabeçalho Ethernet é preenchido, e a seguir é verificado o tamanho da área de dados, que nunca deverá ser menor que 46 bytes. No caso da área de dados possuir menos que 46 bytes então um enchimento com "brancos" (0x20) é executado com o objetivo de atingir o tamanho mínimo de um quadro Ethernet. O próximo passo é remeter o pacote acabado para o controlador de rede, com o uso da função `EnviaPct()`.

A função `EnviaPct()` executa o DMA apropriado visando colocar o pacote na área de memória local do controlador, de forma que este execute os processos necessários ao correto encaminhamento do pacote pela rede.

2.5 O Protocolo ARP (Address Resolution Protocol)

A falta de padronização das camadas física e de enlace de dados pelo TCP/IP pressupõe a utilização de abstração destas camadas pelos protocolos de níveis superiores. Uma das formas desta abstração é a utilização de endereços de nível alto para especificar os hospedeiros pertencentes à rede, independendo totalmente da forma de endereçamento implementada no hardware. Para o TCP/IP, cada hospedeiro possui um endereço IP único, mas de forma alguma, este endereço pertence ao hardware, podendo facilmente ser utilizado por diversos hospedeiros, desde que não ao mesmo tempo. Isto é, o endereço IP é um endereço da camada de rede, que normalmente é implementada em software, e portanto pode ser facilmente alterado. Por outro lado, o endereço de hardware é fixo, como por exemplo, no caso do padrão Ethernet são seis bytes gravados em fábrica diretamente na interface de rede.

Por este motivo é necessária a adoção pela pilha TCP/IP de um mecanismo eficiente de mapeamento de endereços IP em endereços de hardware e, neste sentido, foi padronizado pela RFC 826 [PLU 82] um protocolo de resolução de endereços IP em endereços de hardware denominado ARP.

Conforme citado anteriormente, visando a economia de recursos na implementação em hardware do código resultante, a implementação do protocolo ARP ficou restrita ao cache ARP e ao lado servidor, ou seja, aquele que responde às solicitações de resolução de endereço. Para atingir tal objetivo, o software anteriormente implementado (pingador) sofreu algumas melhorias significativas do ponto de vista de sua funcionalidade.

2.5.1 A Resposta ARP

Partindo do princípio de que o sistema desenvolvido por esta aplicação não executará, em momento algum, o papel de cliente, em qualquer dos níveis do protocolo TCP/IP, fica claro que o comportamento da implementação será apenas reativo, ou seja servidor. Portanto, os aspectos da pilha de protocolos relativos ao lado solicitante (cliente) foram ignorados.

Com este enfoque consegue-se atingir o objetivo de economia de recursos, uma vez que grande parte do código previsto pelos padrões do protocolo não são implementados. Esta estratégia permite que o sistema como um todo tenha seu tamanho reduzido, viabilizando sua integração em hardware.

O ARP possui três blocos funcionais: entrada, saída e gerenciador de cache. Além destes blocos, existe uma estrutura de memória que implementa o cache ARP, onde é baseada toda a funcionalidade do protocolo, sendo, fundamentalmente, uma tabela mantida em memória com o mapeamento de endereços IP para endereços de hardware, com uma entrada para cada hospedeiro da rede com endereço validado pelo ARP. Existem também dados adicionais pertinentes a cada entrada da tabela como, por exemplo, tipo de hardware (Ethernet, Token Ring, etc.), tipo de protocolo de alto nível (IP ou outro qualquer), tamanhos dos endereços utilizados pelos protocolos envolvidos, tempo de vida (TTL) desta entrada entre outros. A figura 2.15 a seguir ilustra o formato do pacote ARP.

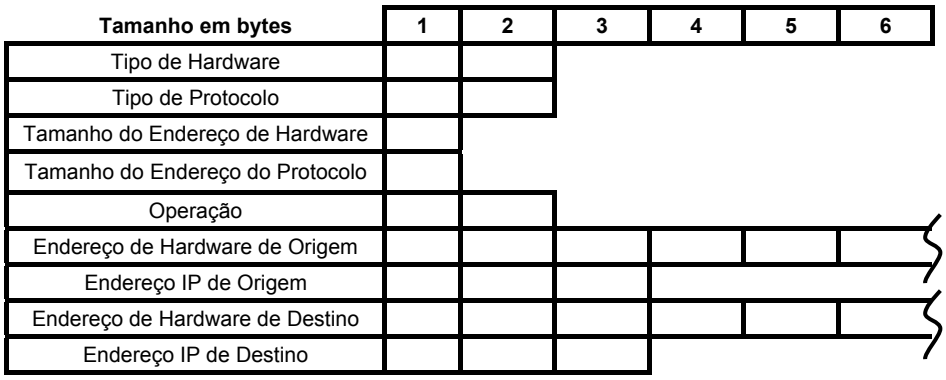

FIGURA 2.15 - Formato dos dados dos protocolos ARP.

Os campos *Tipo de Hardware* e *Tipo de Protocolo* definem qual o hardware utilizado pela interface de rede e qual o protocolo de alto nível que está utilizando o ARP, denotando a diversidade de aplicação do ARP que, desta forma, poderá ser utilizado com qualquer combinação de protocolos de nível dois e três (ISO – OSI) [ISO 94]. A seguir, dois campos de um byte cada especificam o comprimento em bytes do endereço implementado nestes protocolos, definindo, portanto, o tamanho dos quatro últimos campos do quadro. No caso do Ethernet e IP, estes campos possuem, respectivamente, seis e quatro bytes. O campo *Operação* define o tipo da mensagem que pode ser: 1 – Requisição ARP, 2 – Resposta ARP.

Em uma requisição ARP, o hospedeiro origem preenche seus endereços na mensagem e adiciona o endereço IP do destino, o qual deseja descobrir o endereço de hardware, enviando a solicitação para todos os hospedeiros, no nível de enlace de dados. Na resposta à requisição, o hospedeiro de destino preenche seu endereço de hardware, trocando os pares de endereço (origem por destino), altera a operação para resposta ARP e envia um pacote diretamente ao hospedeiro de origem.

A rotina responsável pela resposta à uma solicitação ARP proveniente da rede é executada pela função `ArpResp()` conforme ilustra a figura 2.16 a seguir. Ao receber uma solicitação o chaveador central que encontra-se na função principal do programa faz três verificações sobre o pacote que chega:

1- Se o pacote Ethernet carrega em sua área de dados um pacote ARP, verificando o campo *Tipo de Protocolo* Ethernet.

2- Se o endereço IP de destino contido no pacote ARP confere com o endereço IP local, ou seja se o pacote destina-se ao hospedeiro local.

3- Se o pacote ARP é uma solicitação, tendo em vista que as respostas ARP não são tratadas pelo lado servidor, e o lado cliente não está implementado.

A seguir a função de resposta ARP é chamada se encarregando de todo o processamento dos dados que chegaram, visando a geração de uma resposta adequada ao solicitante.

A primeira ação é a busca do endereço IP do hospedeiro que gerou a solicitação na tabela ARP, visando adicionar o par Endereço IP, endereço MAC na tabela caso não exista. A seguir o tempo de vida (TTL) desta entrada é ajustado em 11 minutos. De

posse dos endereços de origem a resposta ao solicitante pode ser elaborada, e a seguir encaminhada à função `EtherResp()`, para encapsulamento no pacote Ethernet e posterior envio para a rede.

Na figura 2.16, pode-se observar o fluxo de operações envolvido na função de geração de resposta ARP.

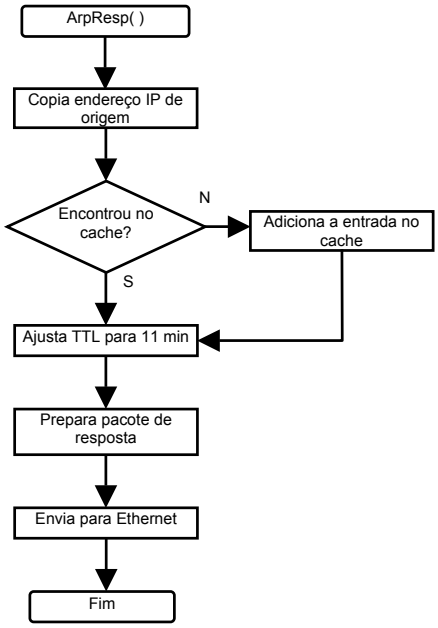

FIGURA 2.16 – Fluxo da função de resposta ARP.

2.5.2 A gerência do cache ARP

A tabela que implementa a memória ARP utilizada por este sistema TCP/IP é bastante simples, tendo em vista as várias opções de campos de dados disponíveis, previstos no padrão. Os campos selecionadas para cada registro que será inserido nesta tabela são:

1- Endereço IP do hospedeiro destino.

2- Endereço MAC do hospedeiro destino

3- TTL desta entrada

A cada solicitação de resolução de endereço que o hospedeiro local receba, como primeira tarefa será feito o acréscimo dos endereços (MAC e IP) do hospedeiro que originou a solicitação no cache, caso ainda não exista tal entrada. Com esta estratégia, a comunicação que se estabelecerá a seguir entre os protocolos de mais alto nível, fluirá sem necessidade de novas resoluções de endereço IP em endereço MAC, tendo em vista que estes já encontram-se resolvidos na tabela local.

O tempo de vida de uma entrada é especificado em 10 minutos, caso esta entrada esteja sendo utilizada regularmente pelos protocolos de nível superior, ou de 2 minutos caso

deixe de ser utilizada. Após estes tempos uma nova solicitação de resolução de endereço deverá ser enviada à rede, visando a atualização da entrada no cache.

Como o sistema implementado não contempla o lado cliente, gerador de solicitações de resolução de endereços, a estratégia utilizada foi colocar o tempo de vida de cada entrada do cache local com um minuto a mais do que o maior tempo de TTL permitido no protocolo ARP.

Segundo o protocolo, o tempo máximo permitido para a validade de uma entrada ARP no cache é de 10 minutos, desde que esta entrada esteja em uso regular pelos protocolos superiores. Portanto cada entrada no hospedeiro local terá o seu TTL fixado em 11 minutos, forçando assim com que os hospedeiros que se conectam ao hospedeiro local sempre sejam obrigados a solicitar a resolução de endereços, liberando o sistema local desta tarefa. Com esta estratégia simples pode-se dispensar completamente o lado cliente do protocolo, atingindo o objetivo de economia de recursos.

A função que desempenha o papel de gerência da tabela ARP local é a `ArpCache()`, que resume suas ações em duas:

1- Busca na tabela por uma entrada já resolvida, e válida, de um determinado endereço IP, para o qual se pretenda enviar um datagrama.

2- Retorno do índice de entrada de uma entrada vazia, ou da entrada mais antiga (caso a tabela já se encontre cheia), caso o endereço IP pretendido ainda não existir como entrada válida.

Esta função é bastante simples, visto que a função de geração de resposta ARP se encarrega do acréscimo das novas entradas que eventualmente cheguem pela rede. A figura 2.17 ilustra o fluxo da função de gerência do cache local.

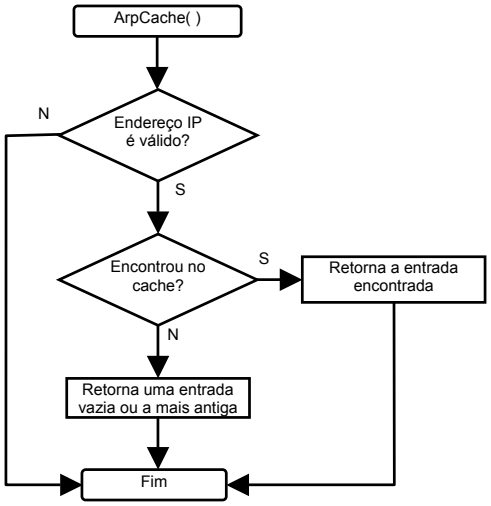

FIGURA 2.17 – Fluxo da função de procura na tabela ARP.

2.6 O Protocolo IP (Internet Protocol)

O protocolo IP é o responsável pelo roteamento de datagramas entre as diversas redes componentes de uma malha de redes, tendo como uma de suas características a falta de confiabilidade, uma vez que não provê mecanismos de detecção ou correção de erros de transmissão para os dados que carrega. Para atingir tal objetivo, o protocolo mantém uma tabela de roteamento em cada hospedeiro e roteador, que, trabalhando de forma cooperativa, encaminham um datagrama de uma origem até o seu destino, mesmo que para isto o datagrama tenha que atravessar diversas redes que não as da origem ou de destino. O datagrama IP é diretamente encapsulado no quadro de nível de enlace de dados, onde, no caso desta implementação será utilizado o padrão Ethernet.

Para prover a confiabilidade necessária às aplicações de rede, como é o caso do trabalho aqui desenvolvido, a próxima camada de protocolo (TCP) fornecerá todos os mecanismos de detecção e correção de erros, como podemos notar no desenvolvimento desta camada mais adiante.

Padronizado pela RFC 791 [POS 81], o protocolo IP apresenta o datagrama com o formato ilustrado na figura 2.18.

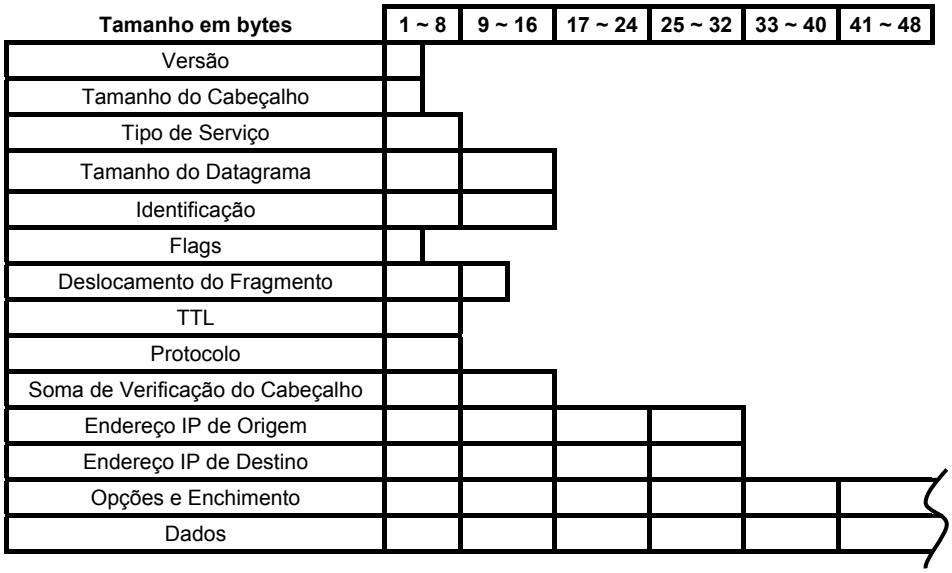

FIGURA 2.18 - Formato do datagrama IP

O campo *Versão* carrega a informação de versão do IP utilizado no datagrama e, caso seja diferente da versão presente no hospedeiro, então o datagrama deve ser descartado (a versão atual é a quatro). *Tamanho do Cabeçalho* especifica o comprimento da área de cabeçalho em múltiplos de 32 bits.

O campo *Tipo de Serviço* especifica como o IP deve tratar o datagrama. Os primeiros três bits formam a precedência onde sete é a maior e zero a normal, permitindo a priorização de datagramas com precedência maior. Os três bits seguintes sugerem como o datagrama deve ser roteado, onde o quarto bit solicita intervalo baixo, o quinto bit Taxa de transferência alta e o sexto bit alta confiabilidade. Nem sempre é possível ao roteador atender a esta demanda, portanto, é comum implementações que não as levem em consideração até por questões de impossibilidade física de atendimento.

Tamanho do Datagrama especifica o tamanho total do datagrama em bytes, *Identificação* identifica um datagrama univocamente e, no caso da necessidade de fragmentação por motivo de uma rede com pequeno MTU (Maximum Transfer Unit), cada fragmento pertencente a um determinado datagrama inicial contém a mesma identificação. O campo *Flags* contém três bits e controla a fragmentação, o primeiro bit não é utilizado, o segundo bit sinaliza se o datagrama pode ou não ser fragmentado, o terceiro bit indica se o fragmento é intermediário ou se é o último fragmento de um datagrama original. O campo *Deslocamento do Fragmento* de 13 bits identifica o deslocamento dos dados originais que o fragmento atual carrega em múltiplos de oito bytes, iniciando em zero até que o último fragmento sinalize um deslocamento igual ao tamanho do datagrama original. O campo *Protocolo* define qual o protocolo está encapsulado nos dados do datagrama, *Soma de Verificação* é um checksum da área de cabeçalho IP, *TTL* sinaliza o tempo máximo que o datagrama pode permanecer transitando entre redes. Cada roteador o decrementa e, atingindo zero o datagrama é descartado. Os campos *Endereço IP de Origem* e *Endereço IP de Destino* especificam a origem e o destino do datagrama e não são alterados, à medida que o datagrama é roteado entre as redes.

O campo *Opções e Enchimento* não é obrigatório, mas existindo, possui comprimento entre um byte até o tamanho máximo do cabeçalho limitado pelo campo *Tamanho do Cabeçalho* e sempre deve ter comprimento múltiplo de 32 bits, daí a necessidade do enchimento. Este campo é utilizado para testes e depuração da rede e, por fim seguem os dados encapsulados.

2.6.1 As Simplificações Utilizadas

Por questões de economia de recursos, a implementação do protocolo IP não segue todas as possibilidades previstas pela RFC, mas apresenta várias limitações que, empregadas de forma consistente, não prejudicam de forma alguma o sistema como um todo.

Por se tratar de um hospedeiro, o sistema não implementa os itens referentes ao roteamento de datagramas, uma vez que tais rotinas são totalmente prescindíveis. Visando atingir o status de roteador, o servidor de dados deveria possuir pelo menos duas interfaces de rede, ou seja, conectar-se fisicamente a pelo menos duas redes, o que não é o escopo deste trabalho. Como resultado desta simplificação muitos recursos de hardware foram economizados, e do ponto de vista do software, muitas rotinas complexas deixaram de ser desenvolvidas. O sistema proposto comporta-se como um hospedeiro servidor de dados, possui uma única interface de rede, e é reativo, isto é, somente responde quando solicitado por um cliente externo. Desta forma o roteamento de datagramas é completamente dispensável.

Outra simplificação utilizada no nível do protocolo IP é quanto à fragmentação de datagramas. Segundo o padrão, quando um datagrama atinge uma rede com MTU menor que o seu tamanho total, o roteador limítrofe deve quebrá-lo em partes menores, a fim de atingir o tamanho correto para atravessar a rede. Desta forma o hospedeiro destino do datagrama partido tem que possuir a capacidade de remontar o datagrama original, com o fim de recuperar seus dados corretamente.

Tendo por premissa que o nível de enlace de dados utilizado no desenvolvimento do trabalho foi o padrão Ethernet, existe a possibilidade de se transportar quadros com até

1500 bytes. Contribuindo com isto a quantidade de dados enviados pelo nível de aplicação (HTTP) deverão normalmente estar bem abaixo disto, então a capacidade de remontagem de datagramas IP foi completamente desconsiderada. Esta posição poderá ser reconsiderada no caso da utilização do sistema proposto em conexões com a Internet, onde redes com pequeno MTU podem ser factíveis, conduzindo desta forma à fragmentação.

2.6.2 A Implementação do Protocolo IP

Por conta das simplificações utilizadas, a implementação do protocolo IP ficou bastante compacta, principalmente devido à característica reativa do sistema. Desta forma, ao receber uma solicitação de um cliente, são executadas algumas verificações de integridade e qualidade do cabeçalho, e a seguir uma resposta é imediatamente montada e remetida ao cliente.

A figura 2.19 ilustra o fluxo da rotina que executa a montagem e despacho do datagrama IP.

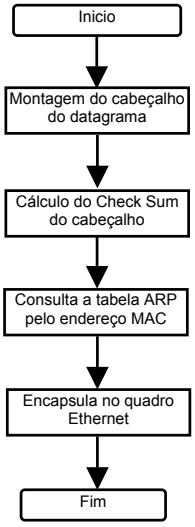

FIGURA 2.19 – O fluxograma da rotina IP

Inicialmente são preparados os campos do cabeçalho com valores fixos, tais como versão, tipo de serviço, os campos pertinentes a fragmentação, TTL, entre outros. Após a preparação inicial do cabeçalho, o campo de soma de verificação é zerado, e a função de cálculo de *CheckSum* é chamada preenchendo o campo com o valor adequado.

Após o cabeçalho estar completamente montado, é executada uma consulta à tabela ARP em memória visando obter o endereço físico do hospedeiro destino do datagrama. Isto é conseguido através da chamada da função `ArpCache()`.

Por fim, o datagrama deverá ser encapsulado no quadro de nível de enlace de dados, com a utilização da função de montagem do cabeçalho Ethernet `EtherResp()`.

2.6.3 O Protocolo ICMP (Internet Control Message Protocol)

O protocolo ICMP é responsável pelo envio de mensagens administrativas na rede, comunicando erros ocorridos durante o roteamento de datagramas, bem como dados de ajuste ou configuração para o hospedeiro originador do datagrama. Os roteadores utilizam o ICMP para a comunicação de datagramas descartados, congestionamentos, TTL excedido entre outros. Os hospedeiros o utilizam para a atualização de rotas na tabela de roteamento, ajuste de horário, teste de acessibilidade etc.

Na interligação em redes, um datagrama flui de roteador em roteador até atingir seu destino final. Se, durante este percurso, houver algum problema que inviabilize a viagem do datagrama, o roteador que tomou a ação de descarte deve enviar um mensagem ICMP indicando a causa do problema ao hospedeiro de origem, para que esse tome as providências necessárias. Deve-se notar que este protocolo não serve para a troca de mensagens de erro entre roteadores, porque, por ser diretamente encapsulado em um datagrama IP, os endereços disponíveis para envio da mensagem são unicamente os de origem e de destino.

O protocolo ICMP é padronizado pela RFC 792 [POS 81a], e os seus dados são diretamente encapsulados em um datagrama IP, com a finalidade de poder atravessar as interligações de rede e atingir o hospedeiro que originou o datagrama com problema, rompendo, desta forma, a barreira da rede local.

A mensagem ICMP possui um cabeçalho e, dependendo do tipo da mensagem, a área de dados com tamanho e conteúdo variável, conforme ilustra a figura 2.20.

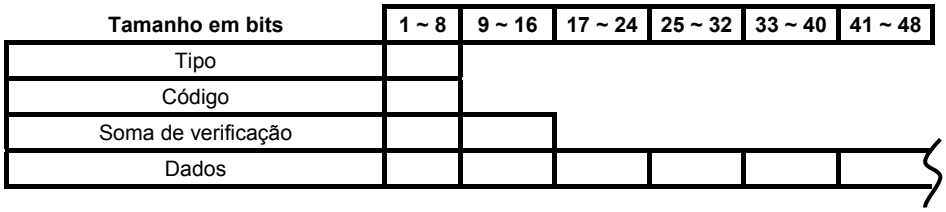

FIGURA 2.20 - Formato da mensagem ICMP

Os campos *Tipo* e *Código* especificam a mensagem carregada pelo quadro, e, a partir daí, a área de dados será preenchida de acordo. O campo *Soma de Verificação* executa um checksum somente sobre a área de dados da mensagem.

Com o intuito de simplificação do código gerado, a maioria das opções de tipos de mensagem, disponíveis no protocolo foram ignoradas, ficando apenas o teste de conectividade via rede, conhecido como eco.

As mensagens do tipo “0” e “8” especificam resposta e requisição de eco, respectivamente, utilizadas para localização de problemas na rede. O campo *Identificador* caracteriza uma determinada solicitação e o campo *Núm. de seq.*, a seqüência das mensagens nesta solicitação. Na área de dados, opcionalmente, a requisição pode colocar qualquer seqüência de caracteres (o tamanho não é fixo) que deverão ser repetidos pela resposta.

2.6.4 A Implementação do Protocolo ICMP

O código resultante do protocolo ICMP é, de forma semelhante ao protocolo IP, bastante simples, como se pode notar na figura 2.21.

A primeira ação tomada pela rotina é a verificação da soma de verificação presente no cabeçalho do datagrama que chega do cliente, com o intuito de detecção de um possível pacote defeituoso. Caso a soma de verificação não confira com a gerada pela função de verificação (`CheckSum()`), o pacote é imediatamente descartado.

Caso a requisição de eco recebida pela rede do cliente esteja íntegra, então a resposta a esta requisição é montada, iniciando pela preparação do cabeçalho ICMP, que é bastante simples. Deve ser tomado cuidado com os campos de identificação e seqüência, com o fim de que o cliente consiga distinguir esta resposta de outras, que possivelmente cheguem ao mesmo tempo de outros hospedeiros, e desta forma tenha sucesso na associação de diferentes solicitações e respostas de eco.

A seguir os dados presentes na requisição de eco que chega devem ser copiados na resposta que está sendo montada, com o fim de envio para o cliente, posto que a área de dados deve ser igual entre requisição e resposta.

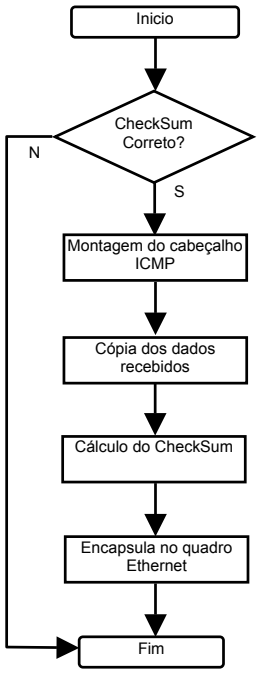

FIGURA 2.21 – Fluxograma da rotina ICMP

Por fim o cálculo da soma de verificação do novo pacote é executada e colocada no campo apropriado. A partir deste momento o pacote ICMP pode ser encapsulado no datagrama IP com o fim de ser remetido, pela interligação de redes, até atingir o cliente que iniciou a solicitação.

2.7 O Protocolo TCP (Transmission Control Protocol)

O TCP é o protocolo mais complexo da pilha TCP/IP, padronizado pela RFC 793 [POS 81b], e apresenta toda a funcionalidade necessária ao gerenciamento de uma conexão fim-a-fim. Localiza-se no nível de transporte, sendo utilizado sempre que as aplicações necessitarem maior confiabilidade na infra-estrutura pouco confiável fornecida pelo IP.

O TCP utiliza um sistema de multiplexação de dados entre as diversas aplicações origem e destino, fornecido pela abstração das portas. Além disto, muitos artifícios são utilizados para que as aplicações nos extremos da comunicação não percam tempo com rotinas voltadas à confiabilidade do ambiente de rede. Exemplos disto são a utilização de esquemas de detecção e correção de erros, características de previsão e prevenção de congestionamentos no tráfego de segmentos TCP pela interligação de redes e a utilização de rotinas de otimização da vazão dos dados pela rede.

Os dados que fluem de forma bidirecional em uma conexão TCP e possuem a característica de fluxo de bits, ou seja, o protocolo não toma conhecimento de estruturas de dados criadas pela aplicação, tudo é remetido de ponta a ponta como um fluxo de bytes contínuo, sem qualquer tipo de demarcadores.

O cabeçalho de um segmento TCP pode ser visto na figura 2.22, e é diretamente encapsulado no datagrama IP, que conterá o valor "6" no campo *Protocolo*, indicando que este datagrama carrega um segmento TCP na sua área de dados.

FIGURA 2.22 - Formato do cabeçalho do TCP

Os campos *Porta de Origem* e *Porta de Destino* contêm, respectivamente, os endereços das portas de protocolo de origem e de destino das aplicações finais. *Seqüência* indica em qual posição dos dados transmitidos pela origem se encontram os dados contidos neste segmento. *Reconhecimento* indica o próximo byte que a origem espera receber a seguir do fluxo de dados que flui do destino para a origem. *Tamanho do Cabeçalho* é um inteiro que contém o tamanho do cabeçalho deste segmento, incluindo as opções, caso existam, e possui o tamanho de quatro bits, sinalizando o tamanho do cabeçalho em múltiplos de 32 bits. A seguir, encontram-se dois campos de seis bits, o primeiro é

reservado para aplicação futura e o segundo contém seis indicadores, conforme a tabela 2.12.

TABELA 2.12 - Significado dos indicadores no cabeçalho TCP

Indicador	Significado
URG	O campo Ponteiro de Urgente é válido
ACK	O campo Reconhecimento é válido
PUSH	Solicita a entrega imediata dos dados à aplicação destino
RESET	Reinicializar a conexão
SYN	Sincronismo de números de seqüência indicado no campo Seqüência
FIN	O emissor encerrou suas transmissões

O TCP no hospedeiro de origem informa ao destino o tamanho máximo de sua área de memória destinada à recepção de dados através do campo *Janela*. A *Soma de Verificação* é feita com a inclusão de um pseudo cabeçalho, onde o campo *Protocolo* conterá o valor seis. O campo *Ponteiro de Urgente*, quando válido, aponta para o final dos dados urgentes que o segmento carrega, indicando que estes dados devem ser imediatamente entregues à aplicação de destino.

O campo *Opções*, quando existir, deve ser múltiplo de 32 bits, daí a necessidade de um enchimento. A principal função deste campo é a negociação para o tamanho máximo do segmento (MSS) utilizado nesta conexão. Finalmente, vem a área de dados do segmento que conterá o tamanho máximo igual ao MSS subtraído do tamanho do cabeçalho TCP.

2.7.1 Simplificações Utilizadas no TCP

Algumas simplificações foram adotadas durante o desenvolvimento do protocolo TCP, visando otimizar a utilização de recursos. O processo de estabelecimento e finalização de uma conexão TCP é bastante complexo, de acordo com o previsto na RFC, e embora não possa ser simplificado de forma expressiva sob pena de fugir do seu escopo, pode ter algumas poucas alterações.

O processo de estabelecimento da conexão, previsto para possuir três etapas, foi mantido inalterado, tendo em vista sua simplicidade. O padrão prevê situações de erro durante esta fase, como por exemplo, a perda de pacotes, que por simplicidade não foram codificadas nesta implementação. Para resolver qualquer impasse durante o estabelecimento da conexão TCP foi implementado um temporizador, que reinicializa a conexão no caso de vencer o seu prazo.

De forma semelhante, o processo de finalização da conexão foi simplificado, sendo previsto apenas a finalização em duas etapas, uma para cada sentido do fluxo de dados. Assim, são utilizados dois pacotes, um de solicitação de finalização e um de reconhecimento. De forma semelhante ao estabelecimento da conexão, a finalização não provê os esquemas de erro estabelecidos pelo RFC.

Após o estabelecimento de uma conexão de forma adequada, o cliente deve fazer uma solicitação HTTP conhecida por *get*, quando o servidor retornará uma página HTTP previamente selecionada, armazenada no servidor. Esta página deve ser simples e de conteúdo estático preferencialmente, não inviabilizando o conteúdo dinâmico, caso necessário. Toda a preocupação gira em torno do fato de que a aplicação resultante deve ser a menor possível, visando economia na área de silício do hardware final.

2.7.2 A Máquina de Estados TCP

De acordo com a RFC 793, uma máquina de estados deve ser implementada visando o gerenciamento da conexão TCP a uma determinada porta. Conforme mencionado anteriormente, são previstas muitas situações por esta máquina, isto é, estados e condições para a troca entre estados. Ainda são previstas situações de erro que devem ser tratadas durante as transições de estados, em vários níveis da máquina de estados TCP.

Com o fim de executar os processos envolvidos na inicialização, manutenção e finalização de uma conexão TCP, a máquina de estados implementada sofreu algumas simplificações, como pode ser notado na figura 2.23.

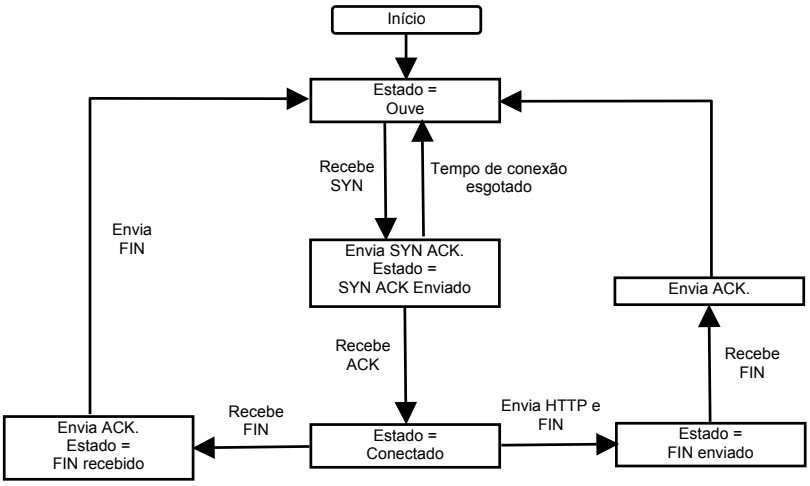

FIGURA 2.23 – Máquina de estados simplificada.

Durante o estabelecimento de uma conexão TCP, é inicializado um temporizador e ajustado em 15 minutos. Durante este período de tempo, o servidor aguarda o recebimento de um pacote SYN. Caso receba o pacote pretendido, então o estado passa a ser Conectado, caso contrário o estado passará a ser Ouve novamente.

Após uma conexão estabelecida com sucesso, durante o estado Conectado, existirá a fase de troca de dados entre cliente e servidor, que de forma bastante simples será uma requisição HTTP *get* enviada pelo cliente, seguida por uma página HTML (Hiper Text Markup Language) retornada pelo servidor. A seguir o servidor envia um segmento de finalização FIN com o fim de encerrar a conexão corrente, no sentido servidor cliente. Após receber o reconhecimento desta finalização o estado da máquina TCP passará a ser FIN enviado, e aguardará o cliente encerrar o seu canal da conexão.

O lado cliente deverá seguir o mesmo processo executado pelo servidor para finalizar o seu canal da conexão. Após o encerramento com sucesso de ambos canais da conexão TCP, a conexão propriamente dita será finalizada, retornando ao estado original Ouve. Desta forma os recursos consumidos por esta conexão serão liberados visando sua utilização por uma nova conexão.

Embora não esteja explícito no diagrama de estados, a qualquer momento a conexão pode ser encerrada utilizando-se o bit de RESET do TCP.

A utilização de conexões rápidas, isto é, conecta, remete a página HTML e finaliza, diminui a necessidade de gerenciamento de grandes quantidades de conexões ativas, reduzindo as necessidades de recursos consumidos pelo servidor, com memória e capacidade de processamento.

2.7.3 O Código TCP

A figura 2.24 a seguir representa o fluxograma do código implementado para a execução do protocolo TCP.

Inicialmente o pseudo cabeçalho TCP é gerado com o fim de cálculo da soma de verificação do pacote que chega, que caso contenha erro, o processo de resposta é abortado imediatamente.

Para o caso de um pacote que chega do cliente correto, é verificada a porta da conexão pretendida. Caso exista, então o fluxo é remetido para a máquina de estados TCP, que se encarrega do restante da lógica envolvida no gerenciamento das conexões ativas. Caso o pacote remetido pelo cliente não pertença a alguma conexão em processo, então deve ser um pedido de estabelecimento de uma nova conexão TCP. Qualquer outra hipótese é inválida ocasionando o fim da execução do processo TCP.

A máquina de estados TCP implementa os processos descritos no item anterior, que disserta especificamente sobre a máquina de estados utilizada na corrente implementação.

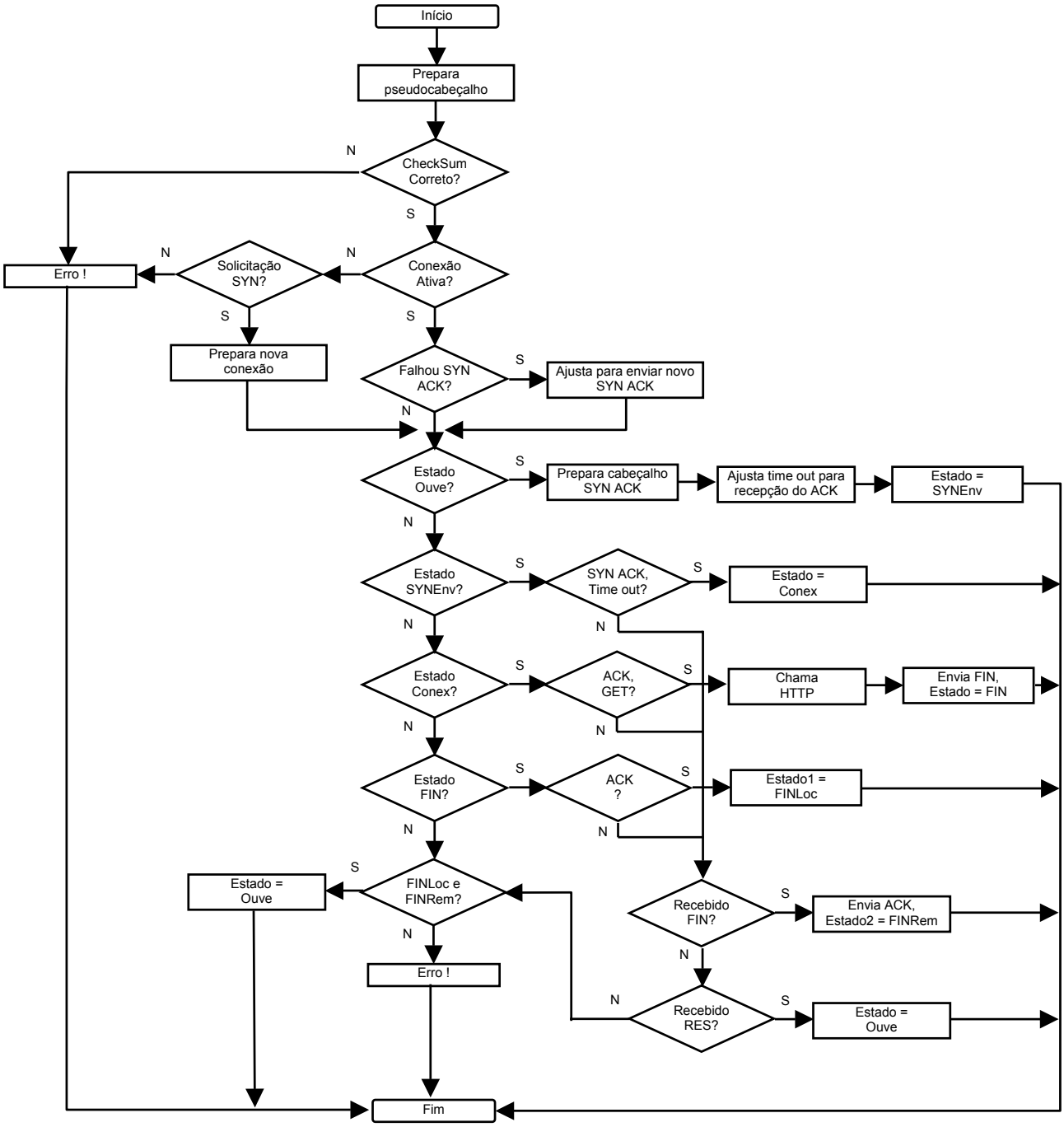

FIGURA 2.24 – Fluxo do protocolo TCP.

3 A Geração de um ASIP

A tendência natural de miniaturização e portabilidade dos sistemas eletrônicos atuais conduz a algumas definições de projeto baseado na orientação do estado da arte da tecnologia. Baseado neste fato, e em vários trabalhos efetuados na tentativa de apontar ou proporcionar ferramentas e métodos para a síntese eficiente de hardware e software para aplicações específicas, este trabalho busca uma solução compacta, de simples emprego e de custo acessível, visando a aplicabilidade praticamente imediata da solução aqui apontada.

Um trabalho que foi utilizado, visando a geração em hardware do sistema aqui proposto, é a análise e síntese de software e hardware dedicado para aplicação específica denominado System as Software and Hardware in Microcontrollers (SASHIMI). Esta ferramenta será utilizada neste trabalho visando a geração de um processador dedicado com um pequeno conjunto de instruções, bem como o respectivo software que executará a aplicação fim, ou seja a pilha TCP/IP.

3.1 A Opção por um Processador Dedicado

Os sistemas embarcados orientam-se pela necessidade de leveza, robustez, consumo, adequabilidade, entre outras características, muitas vezes conflitantes. Baseado no trabalho elaborado por Sérgio Ito [ITO 2000], neste ambiente os microcontroladores dominam o cenário, e a tendência é de que se mantenham neste segmento de mercado, principalmente devido a algumas características interessantes como:

- Devido ao seu normalmente pequeno conjunto de instruções, não existe o custo adicional de hardware exigido pelas instruções desnecessárias à aplicação proposta;

- Fornece um conjunto computacional praticamente completo com unidade de processamento, memória, entradas e saídas, conversores analógico / digital, entre outras funções;

- São pensados de forma a fornecer apenas a quantidade de memória, e outros recursos, necessários às pequenas aplicações em que serão empregados;

- Apresentam programabilidade simplificada, principalmente devido a uma ampla gama de ferramentas de auxílio ao programador, proporcionando facilidades de escolha de linguagem de programação como Assembler, C, Pascal, entre outras;

- Com a pressão do mercado, novas funções são constantemente adicionadas ao hardware, simplificando o desempenho de tarefas complexas como por exemplo a adição de núcleos de processamento digital de sinais (DSP), ou a possibilidade de utilização de características muito desejáveis na tecnologia embarcada, como o controle do consumo de energia do dispositivo.

Mas embora sejam muitas as qualidades apresentadas pelos microcontroladores, uma característica é fundamental: o custo reduzido. Este custo é derivado principalmente da adequabilidade do conjunto hardware / software à aplicação fim, e dos volumes envolvidos.

Considerando-se todas as vantagens apresentadas quando da utilização de microcontroladores em sistemas embarcados, deve-se levar em conta uma importante característica que pode ser agregada ao dispositivo visando otimizar ainda mais seu conjunto de vantagens, ou seja, a implementação somente das instruções realmente necessárias à aplicação. Com este artifício gera-se um processador com instruções específicas para a aplicação (Application Specific Instuction set Processor - ASIP), que apresentará o menor custo do conjunto hardware, embora o desenvolvimento de software fique bastante limitado, uma vez que o caráter de aplicabilidade ampla de um conjunto completo de instruções deixa de existir. Devido a este fato, deve ser bastante criteriosa a definição do conjunto de instruções que será implementado no sistema.

Durante a fase de elaboração do software aplicativo para o sistema embarcado com a utilização de ASIP's, é fundamental a utilização de um compilador adequado ao conjunto de instruções proporcionado pelo processador, recaindo sobre este ponto a principal desvantagem desta abordagem. Como conseqüência, o programador fica basicamente limitado à elaboração de programas em Assembler, abrindo mão da reutilização, e com a desvantagem da difícil manutenção do código resultante.

Conforme [KRE 97], quatro abordagens de solução podem ser implementadas na tentativa de resolver os problemas específicos gerados por uma determinada aplicação fim. A adequabilidade de uma ou outra das possíveis soluções depende de uma análise, por muitas vezes complexa do ambiente da aplicação, de sua exeqüibilidade, custo, disponibilidade de ferramentas, prazos, entre outros tantos fatores que por ventura surjam durante o desenvolvimento do sistema.

A primeira aproximação é o desenvolvimento de um circuito integrado específico para a aplicação fim, onde a principal desvantagem é o custo de desenvolvimento, mas por outro lado apresenta, sem dúvida, o melhor desempenho.

A segunda abordagem é a elaboração de um processador específico para a aplicação, que, conforme já discutido, possui um conjunto de instruções reduzido e totalmente voltado para a aplicação fim. Com esta abordagem o custo de desenvolvimento do hardware é grande, tendo em vista a sua adequação à aplicação, mas o custo final do sistema será reduzido, uma vez que os elementos desnecessários não serão implementados, reduzindo assim o custo do hardware final.

Outra possibilidade é o projeto resultante do projeto conjunto de software e hardware onde ao processador central será adicionado algum hardware específico voltado para a aplicação fim. Com esta aproximação, o desempenho final do sistema será próximo do processador de aplicação específica, com a vantagem da programação fácil.

Por fim, há a solução clássica do processador de propósito geral com a condução do problema totalmente orientada ao software. As vantagens do ponto de vista de programabilidade são grandes e poderosas, mas com o custo gerado pelo hardware grande e, por muitas vezes subutilizado desta aproximação, é o maior que qualquer outra abordagem anteriormente discutida.

No trabalho aqui apresentado, foi utilizada uma ferramenta de software com o propósito de gerar o código VHDL para a integração em hardware do processador de aplicação específica TCP/IP. Esta ferramenta foi desenvolvida por Sérgio Ito e chama-se SASHIMI [ITO 2000]. Esta ferramenta foi escolhida devido à sua facilidade de aplicação ao ambiente aqui proposto, ou seja, a partir do código gerado em um computador utilizando a linguagem de programação C, foi simples a tradução para o Java, que serve de fonte de entrada para o SASHIMI. Em adição a isto, a ferramenta gera o hardware descrito em VHDL de um processador de aplicação específica, conforme a necessidade da aplicação fim. Além disto, o software aplicativo que executará no processador gerado anteriormente, é também fornecido pela ferramenta em questão.

Com todas as facilidades apresentadas, o SASHIMI tornou-se uma solução bastante atraente visando a geração do hardware pretendido por este trabalho. Caso se leve em consideração os esforços empregados por qualquer outro meio de desenvolvimento de hardware e software dedicado a este fim, fica flagrante a aplicabilidade imediata desta ferramenta, bem como a velocidade e simplicidade no fornecimento de resultados finais.

3.2 A Ferramenta SASHIMI

O primeiro estágio do trabalho foi o desenvolvimento da pilha de protocolos formadores do TCP/IP com o auxílio da linguagem C, e do hardware do PC AT. O passo seguinte foi a simplificação do código em C, resultante de uma abordagem já funcional e depurada no ambiente de rede Ethernet para um programa, ainda em C, mas com um conjunto de instruções restrito àquelas que a ferramenta SASHIMI implementa sem restrições. Por exemplo, o Java utilizado pelo SASHIMI não suporta estruturas de dados da forma como são implementadas no C através do comando *struct*, bem como operações com ponto flutuante.

Esta etapa foi orientada no sentido de remover todas as instruções, que de uma forma ou outra, viessem a impossibilitar a utilização da ferramenta em questão. Outro aspecto neste sentido foi a eliminação de todas as funções de bibliotecas externas fornecidas pela linguagem C disponíveis através do comando *include*, e a sua conseqüente codificação diretamente no corpo do programa principal.

A partir deste ponto, a tradução do código em C para o Java compatível com o SASHIMI foi simples, praticamente direta, uma vez que as linguagens em questão são muito semelhantes do ponto de vista sintático.

Com a utilização da ferramenta SASHIMI, tomando por base o programa Java como entrada, foi gerada uma descrição VHDL de um processador específico para a aplicação, que apresenta ao usuário um conjunto reduzido de instruções completamente voltadas para a aplicação fim. Este processador gerado em um FPGA [ALT 2001a] apresenta algumas características conforme discutido a seguir.

A arquitetura é composta de três unidades distintas com as funções de seqüenciamento do fluxo de programa, gerenciamento da pilha e cálculo lógico e aritmético sobre a pilha. Apresenta uma arquitetura do tipo de pilha com barramentos distintos para memória de programa e memória de dados. Possui ainda portas de entrada e saída, bem

como um temporizador e dois níveis de interrupção. A figura 3.1 ilustra a arquitetura utilizada no microcontrolador descrito pela ferramenta SASHIMI.

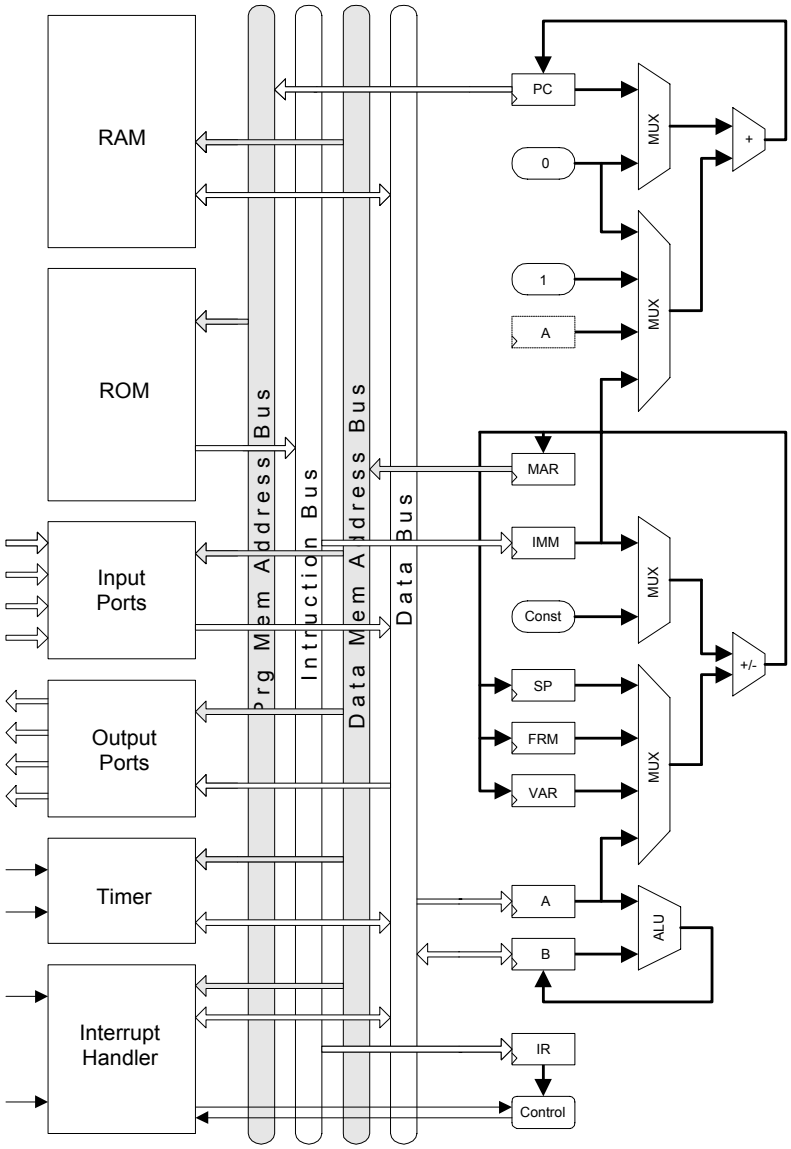

FIGURA 3.1 – Arquitetura do microcontrolador FemtoJava.

As instruções geradas pela ferramenta SASHIMI são um subconjunto das instruções disponíveis em uma máquina virtual Java normal, conforme ilustra a tabela 3.1. Das 68 instruções implementadas originalmente no processador Java, somente as realmente necessárias à aplicação fim serão disponibilizadas pelo hardware para serem utilizadas pelo software aplicativo, também fornecido pela ferramenta.

Ao ser compilado o código Java será gerado um arquivo com bytecodes Java que terá a extensão .class. A partir deste arquivo é feita uma análise dos bytecodes realmente utilizados pela aplicação, e a partir desta definição do conjunto de instruções é gerado o processador específico para esta aplicação fim.

TABELA 3.1 – Conjunto de instruções implementadas no SASHIMI

Tipos de Instrução	Mnemônicos
Lógica Aritmética	iadd, isub, imul, ineg, ishr, ishl, iushr, iand, ior, ixor
Controle de Fluxo	goto, ifeq, ifne, iflt, ifge, ifgt, ifle, if_icmpeq, if_icmpne, if_icmplt, if_icmpge, if_icmpgt, if_icmple, return, invokestatic
Operações de Pilha	iconst_m1, iconst_0, iconst_1, iconst_2, iconst_3, iconst_4, iconst_5, bipush, pop, pop2, dup, dup_x1, dup_x2, dup2, dup2_x1, swap
Load/Store	Iload, iload_0, iload_1, iload_2, iload_3, istore, istore_0, istore_1, istore_2, istore_3
Array	Iaload, baload, caload, saload, iastore, bastore, castore, sastore, arraylength
Estendido	load_idx, store_idx, sleep
Outras	nop, iinc, getstatic, putstatic

3.3 A Implementação

O código implementado em linguagem C foi testado em um ambiente de rede de computadores com PCs, apresentando resultados positivos do ponto de vista de conectividade e resposta. O ambiente utilizado foi o apresentado na figura 3.2.

FIGURA 3.2 – Ambiente de teste.

Neste ambiente foram depurados os problemas de conexão entre hospedeiros, conforme prevê a padronização da pilha de protocolos TCP/IP.

Os protocolos foram sendo desenvolvidos, aplicados e depurados um a um de acordo com sua posição relativa na pilha, iniciando pelo protocolo de nível de enlace de dados, o IEEE 802.2. Embora este protocolo esteja diretamente integrado no hardware do controlador Ethernet, dispensando a sua codificação pelo usuário, existem outras tarefas que devem ser executadas pelo código desenvolvido, visando ativar o controlador, e após sua correta ativação, manter o fluxo de pacotes nos dois sentidos da rede.

Após a ativação do controlador Ethernet, o primeiro protocolo codificado foi o ARP, responsável pelo descobrimento na rede dos endereços de hardware de um dado hospedeiro. A seguir o protocolo de nível de rede foi codificado, o IP, visto que sobre este nível se estabelece toda a troca de informação entre hospedeiros, na forma de datagramas.

Com a infra-estrutura de troca de dados funcional, foi codificado de forma simplificada o protocolo ICMP, responsável pelo utilitário de descoberta de hospedeiros na rede PING. Este utilitário é também utilizado para a depuração de problemas de conectividade, sendo bastante útil durante todo o processo de desenvolvimento e testes deste trabalho.

Ainda sobre a infra-estrutura definida pelo protocolo IP, foi elaborado o protocolo TCP, responsável por toda a confiabilidade na troca de dados entre aplicações. Sobre o TCP foi elaborado um protocolo bastante simples, no nível de aplicação, baseado no HTTP, que tem por objetivo retornar ao cliente uma pequena página HTML.

Com toda a pilha de protocolos funcional, pôde ser medido o tempo de resposta do sistema a uma requisição do cliente, que de forma esquemática, pode ser observada na figura 3.3 a seguir.

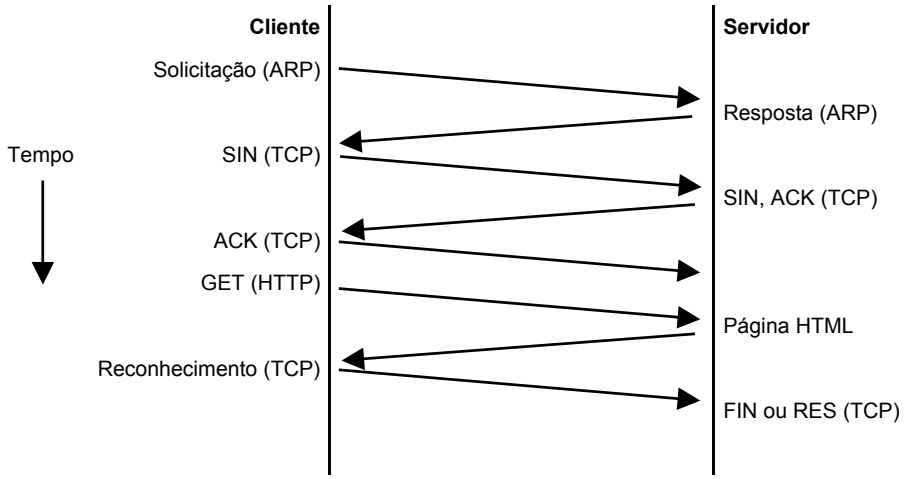

FIGURA 3.3 – Troca de dados entre cliente e servidor.

O fluxo inicia com uma tentativa do cliente em resolver o endereço MAC do servidor baseado em seu endereço IP. O protocolo utilizado neste instante é o ARP, que retorna do servidor uma resposta contendo o seu endereço de hardware.

Com o endereço MAC resolvido o cliente solicita o estabelecimento de uma conexão TCP com o servidor, utilizando a seqüência de três pacotes, conforme esplanado no capítulo anterior. Caso a conexão seja corretamente estabelecida entre os dois hospedeiros, inicia-se a fase de troca de dados de aplicação, que no presente trabalho foi escolhido o protocolo HTTP. Neste ponto o cliente envia ao servidor a requisição de uma página HTML, e o servidor retorna a página solicitada, aguardando o recebimento correto dos dados através dos esquemas previstos no TCP para detecção e correção de erros.

Com a confirmação do recebimento correto dos dados enviados, o servidor imediatamente inicia a seqüência de desconexão do protocolo TCP, liberando desta forma recursos de para uma possível conexão de outro cliente.

O código implementado em C foi pensado de forma a ser o menor possível, sem afetar o desempenho ou a funcionalidade da pilha de protocolos. O processo de conexão, troca de dados e desconexão foi medido e apresentou-se na faixa de 1 a 2 segundos. Este tempo de resposta do sistema pode ser sobremaneira melhorado caso seja implementada a possibilidade do servidor processar mais de um pacote de dados de forma paralela. Na atual configuração, para cada pacote que chega ao servidor, é executado seu processamento de forma completa, elaborada uma resposta, caso seja pertinente, e remetida ao cliente. Isto implica em que, chegando dois pacotes de dados, um em seguida do outro, o segundo será desprezado, enquanto o servidor processa o primeiro, ficando a cargo do cliente reenviá-lo. Esta aproximação, embora tenha liberado valiosos recursos de hardware e software, apresentou a desvantagem de adicionar um pequeno atraso no fluxo de troca de informações. Pode-se estimar que, caso fosse implementado o processamento paralelo de pacotes, o tempo de resposta do sistema cairia para algo em torno de 0,1 a 0,5 segundo, baseado nas observações feitas em softwares de depuração de redes conhecidos por *sniffers*.

Visando atingir o objetivo de consumir a menor área de silício possível no processador TCP/IP resultante, o código gerado em C manteve-se o menor possível. Como comparação pode-se observar a tabela 3.2, onde é mostrado o tamanho de código fonte de uma aplicação desenvolvida para ser utilizada em ambientes onde o meio de comunicação são rádios [ACK 92] e o código elaborado neste trabalho. A implementação TCP/IP desenvolvida para ser utilizada com rádios foi escolhida como parâmetro de comparação porque foi totalmente elaborada em C, assim como o código aqui desenvolvido.

TABELA 3.2 – Comparação de códigos fonte.

Aplicação	ARP	IP	ICMP	TCP	Total
TCP/IP on Pocket Radio	9 kB	16 kB	7 kB	44 kB	76 kB
TCP/IP em C	3 kB	2 kB	2 kB	7 kB	14 kB

Nesta tabela são comparados apenas os códigos que executam a lógica envolvida no processo de elaboração do protocolo em questão, deixando-se de lado todas as funções e bibliotecas acessórias. Um item importante a ser levado em consideração na comparação entre tamanhos de código é que na implementação em C do processador TCP/IP não existe o tratamento do lado cliente do sistema, conforme anteriormente descrito, redundando em uma diminuição significativa do código resultante. Como uma métrica simples deste fato podemos tomar o código de tratamento do protocolo ARP, por exemplo, do TCP/IP on Pocket Radio com 9kB de tamanho, que trata tanto o lado cliente quanto servidor, em comparação com o TCP/IP codificado em C deste trabalho com apenas 3kB, que trata somente o lado servidor.

TABELA 3.3 – Comparação de códigos executáveis.

Aplicação	ARP	IP	ICMP	TCP	Total
TCP/IP em C	10 kB	10 kB	10 kB	12 kB	26 kB
TCP/IP em Java	0,4 kB	0,2 kB	0,2 kB	1,2 kB	2 kB

Observando a tabela 3.3 fica evidente a redução do código obtida quando da implementação em bytecodes Java, o tamanho do código que executa as funções dos protocolos envolvidos na aplicação fim, quando desenvolvido em C obteve um tamanho de 26 kB, quando da sua tradução para Java o mesmo código reduziu para 2 kB. A comparação não pode ser feita com muita precisão porque o compilador C introduz uma quantidade considerável de código, sem o controle do usuário, no executável resultante, mas mesmo assim a redução é notável.

Com o código elaborado em C pronto, foi feita a sua simplificação, com a intenção de facilitar, mais tarde, a tradução para a linguagem Java. Foram retiradas todas as estruturas de dados implementadas originalmente, transformando-as em vetores simples. Todas as funções que operavam sobre inteiros de 16 bits foram simplificadas para passar a operar sobre inteiros de 8 bits. Finalmente, algumas funções disponíveis nas bibliotecas do C foram codificadas diretamente sobre o corpo principal do programa.

Partindo do código C simplificado, foi iniciada a tradução para o Java, linguagem que serviu de entrada para a ferramenta SASHIMI. Esta tradução foi praticamente direta, visto que ambas linguagens são bastante semelhantes do ponto de vista sintático. Os cuidados que foram tomados durante esta fase foram, principalmente, devido às diferentes características dos hardwares utilizados, ou seja o PC e o microcontrolador de aplicação específica.

O sistema de interrupções do PC e o sistema de interrupções do microcontrolador possuem características próprias, e foram tratados de forma particular, evitando a tradução direta. Outro aspecto diferenciado é a forma de acesso ao barramento ISA do controlador Ethernet, que devido a particularidades dos dois sistemas, teve formas de codificação bastante diferenciadas.

A figura 3.4 ilustra o fluxo de ações executadas com o fim de gerar a descrição VHDL do microcontrolador de aplicação específica, utilizando-se a ferramenta SASHIMI, bem como o programa de simulação e testes fornecido pela Altera chamado Max + Plus II [ALT 2001].

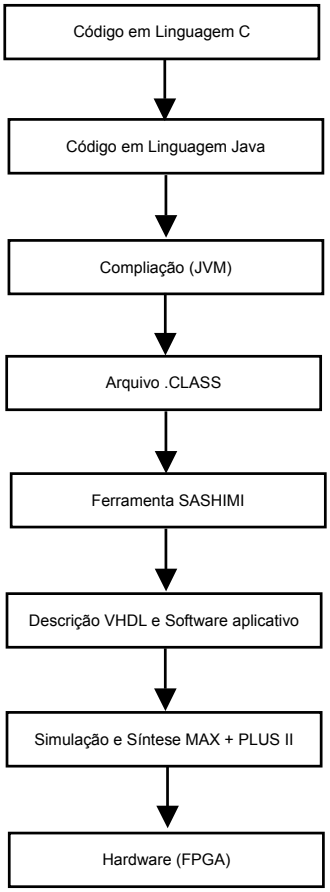

FIGURA 3.4 – Fluxo de síntese do microcontrolador TCP/IP

Após a tradução para o Java do programa fonte, foi executada sua compilação para uma Máquina Virtual Java, que gerou um arquivo com a extensão ".class". Com base neste arquivo a ferramenta SASHIMI produz a descrição VHDL do hardware do microcontrolador de aplicação específica, implementando no seu decodificador de instruções apenas as instruções necessárias para o programa aplicativo que será executado por este sistema. A ferramenta produz também o código do programa aplicativo que deverá ser armazenado na memória do microcontrolador.

O código VHDL serviu de entrada para a segunda ferramenta utilizada no processo de geração do microcontrolador, Max + Plus II que executa a simulação e síntese do hardware. O resultado desta síntese pode ser observado na tabela 3.3.

TABELA 3.4 – Resultados da síntese do hardware

Aplicação	# Cél. lógicas	% FPGA	Freq. (MHz)	# Intruções
FemtoJava	1979	85	3,93	68
Proc. TCP/IP	1764	61	8,71	42
Família: Flex 10k (Altera) - Dispositivo: EPF10K50VRC240-1				

Para efeito de comparação, está sendo mostrada a ocupação do FPGA que integra o microcontrolador com todas as 68 instruções disponibilizadas, chamado de Femto Java [ITO 2001] e a seguir o mesmo microcontrolador com apenas as instruções necessárias à aplicação TCP/IP. Nota-se a pequena redução no número de células lógicas ocupadas no FPGA utilizado para a síntese. Isto se verifica porque o decodificador de instruções do processador TCP/IP possui apenas 42 instruções diferentes frente às 68 do microcontrolador Femto Java.

Outro aspecto, não menos importante, da implementação é a velocidade do circuito, denotada pela sua freqüência de relógio. Nota-se que o processador TCP/IP pode processar a praticamente o dobro da velocidade do microcontrolador Femto Java original. Esta comparação pode ser feita diretamente, porque o hardware componente dos dois controladores são basicamente idênticos, diferindo apenas na quantidade de instruções disponibilizadas ao software aplicativo.

4 Conclusão

A troca de dados entre equipamentos de controle e proteção elétrica, empregados em subestações de energia elétrica, apresenta muitas opções do ponto de vista dos meios físicos de conexão em rede, bem como de protocolos de comunicação. Embora alguns padrões internacionais sejam publicados e de alguma forma aceitos, o fato é que, do ponto de vista prático, muito pouca padronização é encontrada nos diversos equipamentos ofertados no mercado.

Com a intenção de apresentar uma solução de padronização para este ambiente específico de comunicação de dados, este trabalho implementou uma solução de conexão via rede de equipamentos que necessitem de uma interface de comunicação, com a intenção de troca de dados de forma local, com redes locais, ou remota, através do emprego de redes de longo alcance.

O ponto central do trabalho foi o desenvolvimento de uma peça de hardware que, uma vez conectada no equipamento gerador de dados, facilite a conexão e o estabelecimento de uma sessão de troca de dados. Para atingir tal simplicidade de conexão, é necessária a utilização de padrões muito bem aceitos e difundidos pelo mercado mundial de sistemas de troca de informação. Um padrão bastante aceito para protocolo de comunicação de dados é o TCP/IP, que foi utilizado para este desenvolvimento.

A pilha de protocolos TCP/IP apresenta uma gama de protocolos com fins específicos, que devem ser implementados visando compatibilidade com os sistemas existentes. Mas, por outro lado, embora a quantidade de protocolos da pilha seja bastante extensa, nem todos necessitam ser implementados, simplificando de forma acentuada a implementação. Mesmo os protocolos que devam ser implementados, visando o correto funcionamento da comunicação de dados, podem, obedecendo alguns critérios, ser simplificados. Desta forma conseguiu-se a implementação de uma pilha de protocolos, supostamente extensa, de uma forma bastante simples.

A intenção principal da simplicidade da pilha de protocolos foi a sua integração em uma peça de hardware, chamada processador TCP/IP, que possui a limitação do tamanho máximo da área de silício a ser utilizada. Outro aspecto, não menos importante desta simplicidade, é o custo total do processador, que se implementado de forma completa, poderia se tornar proibitivo, inviabilizando sua aplicação.

Com as observações feitas durante o desenvolvimento do trabalho prático, ficou clara a possibilidade de execução de um sistema dedicado ao processamento da pilha de protocolos TCP/IP baseado totalmente em hardware. Desta forma, este sistema monolítico libera o ambiente de geração e de consumo de dados para a execução de outras tarefas, que não o processamento de datagramas ou pacotes provenientes da rede local.

O processador TCP/IP em conjunto com o controlador Ethernet são peças de custo bastante acessível, baseado principalmente na sua simplicidade. Possuem uma pequena área total, simplificando sua portabilidade, em ambientes onde o tamanho e peso sejam decisivos. Apresentam excelentes características de conectividade, baseado no nível físico (Ethernet) e a pilha de protocolos utilizado para a troca de dados. Com todas as características aqui descritas o trabalho atingiu os objetivos inicialmente propostos, tornando viável a sua aplicação no ambiente de controle a que se destina, bem como em outros ambientes onde necessidades semelhantes de troca de informações via rede se apresentem.

4.1 Desenvolvimentos Futuros

Outras abordagens do processador TCP/IP podem ser elaboradas visando aumentar sua simplicidade, diminuir seu custo e tamanho, bem como facilitar sua aplicabilidade.

Um exemplo neste sentido é a utilização de um microcontrolador comercial acessando diretamente o controlador de rede Ethernet, que apresentará algumas vantagens do ponto de vista de custo final e de ambiente desenvolvimento, visto que estes microcontroladores são muito difundidos, baratos e apresentam muitas ferramentas para desenvolvimento. No aspecto do desempenho reside a dúvida quanto à viabilidade de aplicação. Entretanto, de uma forma geral, os microcontroladores atuais estão apresentando características de processamento cada vez mais velozes, o que pode viabilizar este tipo de desenvolvimento.

Um aspecto negativo desta abordagem é que muito pouco, senão tão somente a lógica empregada, poderá ser aproveitada do trabalho aqui elaborado, visando simplificar o desenvolvimento do programa aplicativo do microcontrolador. Isto deve-se principalmente, devido às diferenças muito acentuadas nos hardwares propostos, ou seja, o controlador TCP/IP sintetizado em hardware, e o microprocessador interno ao microcontrolador empregado.

Outro desenvolvimento a ser elaborado é, de forma sistemática e bem fundamentada, a eliminação das simplificações da pilha de protocolos utilizados no sistema aqui proposto. Estas simplificações são necessárias atualmente visando principalmente a diminuição da área de silício utilizada durante a integração em hardware do processador TCP/IP. Mas estas simplificações, de uma forma ou outra, afetam o desempenho geral do sistema, deixando-o limitado quanto às suas funções de conectividade em rede. Esta elaboração será muito crítica, tendo em vista a sua complexidade, ou seja, qual das simplificações são realmente elegíveis como necessárias para o perfeito funcionamento do sistema como um todo. Até que ponto devem ser implementadas, completas como prevêem os padrões, ou apenas as funções mais freqüentemente utilizadas? Deve-se ter em mente que a área de silício ocupada pelo processador resultante deve ser mantida a menor possível.

Sob o ponto de vista de conectividade em rede, o padrão físico adotado neste trabalho possui ampla gama de aplicação na maioria dos ambientes em que o protocolo TCP/IP seja utilizado. Mas uma alteração, no sentido de retirar o controlador Ethernet e acrescentar no seu lugar um modem (UART), visando a conexão via linha de comunicação de dados, privada ou dedicada, dará uma outra dimensão ao aspecto da

interligação em redes de longa distância. Esta abordagem implicará no acréscimo de um protocolo específico para o nível de enlace de dados, que na pilha TCP/IP é conhecido por PPP (*Point to Point Protocol*), e a conseqüente retirada do protocolo de enlace Ethernet. O protocolo deste nível utilizado pelo controlador Ethernet está integrado em hardware no próprio controlador, liberando o processador TCP/IP desta tarefa. Por outro lado, no caso do protocolo PPP, a tarefa de controlar o nível de enlace ficará a cargo do processador TCP/IP, adicionando mais um nível na sua pilha de protocolos, exigindo assim, mais recursos de hardware e área de silício, cobrando seu custo no desempenho geral do sistema. Considerando-se que as velocidades de transmissão de dados utilizadas pelos modems são, de maneira geral, bastante inferiores às velocidades apresentadas por uma rede Ethernet, pode se tornar viável este tipo de aproximação.

Bibliografia

[ACK 92] ACKERMANN, John R. **Getting Started with TCP/IP on Pocket Radio.** 1992. Disponível em: <http://sr2tcp.ampr.torun.pl/packet/intro/index.html>. Acesso em: 06 fev. 2002.

[ALT 2001] ALTERA. **MAX+PLUS II Software Overview.** 2001. Disponível em: <http://www.altera.com/>. Acesso em: 06 dez. 2001.

[ALT 2001a] ALTERA. **FLEX 10K Embedded Programmable Logic Family Data Sheet.** V 4.1. 2001. Disponível em: <http://www.altera.com/>. Acesso em: 07 dez. 2001.

[BEC 2000] BECK IPC GMBH. **Hardware Manual:** @CHIP SC12. 2000. Disponível em: <http://www.bcl-online.de/>. Acesso em: 25 out. 2001.

[COM 98] COMMER, Douglas E. **Interligação em Rede com TCP/IP.** Rio de Janeiro: Campus, 1998. 670p.

[COM 99] COMMER, Douglas E.; STEVENS, David L. **Interligação em Rede com TCP/IP.** Rio de Janeiro: Campus, 1999. v.2.

[DAV 97] DAVICOM SEMICONDUCTOR INC. **DM9008 ISA/Plug and Play Super Ethernet Controler.** 1997. Disponível em: <http://davicom.com.tw>. Acesso em: 20 fev. 2001.

[GEN 2001] GENERAL ELECTRIC. **T60 Transformer Management Relay.** 2001. Disponível em: <http://www.geindustrial.com/industrialsystems/pm/>. Acesso em: 24 out. 2001.

[IEE 96] INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **802.3 – CSMA/CD Access Method.** [S. l., 1996]. 1268p.

[INT 95] INTERNATIONAL ELECTROTECHNICAL COMMISSION. **Telecontrol Equipament and Systems – Part 5:** Transmission Protocols – Section 101: Companion Standart for Basic telecontrol Tasks. Genebra, 1995. 181p.

[ISO 94] INTERNATIONAL ORGANIZATION FOR STANDARTIZATION / INTERNATIONAL ELECTROTECHNICAL COMMISSION. **Open Systems Interconnection – Basic Reference Model:** The Basic Model, ISO/IEC 7498-1. [S.l.], 1994. 59p.

[ITO 2000] ITO, Sérgio Akira. **Projeto de Aplicações Específicas com Microcontroladores Java Dedicados.** 2000. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[ITO 2001] ITO, Sérgio Akira; JACOBI, Ricardo Pezzuol. **Making Java Work for Microcontroller Applications.** IEEE, 2001. 12p.

[JAM 98] JAMMEL, Arhtar; STUEMPFLE, Matthias; JIANG, Daniel; FUCHS, Axel. Web on Wheels: Toward Internet- Enabled Cars. **Computer,** Piscataway, v.31, n.1, p. 69-76, Jan. 1998.

[KER 86] KERNIGHAN, Brian W.; RITCHIE, Dennis M. **C a Linguagem de Programação.** Rio de Janeiro: Campus, 1986. 208p.

[KRE 97] KREUTZ, Márcio Eduardo. **Geração de Processador para Aplicação Específica.** 1997. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[LIG 00] LIGHTNER ENGINEERING. **How to Build a Pico Web Project. Version 2.00.** 2000. Disponível em: <http://www.picoweb.net/>. Acesso em: 20 fev. 2001.

[LOE 99] LOEWEN, Myron. **AN724 Using PICMicro MCUs to connect to Internet via PPP.** 1999. Disponível em: <http://www.microchip.com/>. Acesso em: 20 fev. 2001.

[MOD 96] MODICON INC. **Modicon Modbus Protocol Reference Guide.** 1996. Disponível em: <http://www.modicon.com/techpubs/TechPubNew/>. Acesso em 19 nov. 2001.

[MUR 98] MURHAMMER, Martin W. et al. **TCP/IP Tutorial and Technical Overview.** 1998. Disponível em: <http://redbooks.ibm.com>. Acesso em: 10 dez. 2001.

[NAT 95] NATIONAL SEMICONDUCTOR CORPORATION. **DP83902A ST-NIC. Serial Network Inteface Controller for Twisted Pair.** 1995. Disponível em: <http://national.com>. Acesso em: 10 abr. 2000.

[PLU 82] PLUMMER, D.C. **Ethernet Address Resolution Protocol:** Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware, RFC 826. 1982. Disponível em: <http://sunsite.auc.dk/RFC/>. Acesso em: 28 fev. 2000.

[POS 81] POSTEL, J. **Internet Protocol,** RFC 791. 1981. Disponível em: <http://sunsite.auc.dk/RFC/>. Acesso em: 28 fev. 2000.

[POS 81a] POSTEL, J. **Internet Control Message Protocol,** RFC 792. 1981. Disponível em: <http://sunsite.auc.dk/RFC/>. Acesso em: 28 fev. 2000.

[POS 81b] POSTEL, J. **Transmission Control Protocol.** 1981. Disponível em: <http://sunsite.auc.dk/RFC/>. Acesso em: 23 fev. 2001.

[PRE 99] PREDKO, Myke. **PC Ph. D. Inside PC Interfacing.** New York: McGraw-Hill, 1999. 954p. p. 36 - 40.

[PRO 99] PROFIBUS NUTZERORGANISATION. **Profibus Technical Description.** 1999. Disponível em: <http://www.profibus.com/downloads.html>. Acesso em: 18 abr. 2002.

[REA 95] REALTEK SEMI-CONDUCTOR CO. **RTL8019 Realtek Full-Duplex Ethernet Controller with Plug and Play Function.** 1995. Disponível em: <http://www.realtek.com.tw/>. Acesso em: 20 fev. 2001.

[SCE 2000] SCENIX SEMICONDUCTOR INC. **AN35 SX-Stack: Internet Protocols Implementation.** 2000. Disponível em: <http://www.scenix.com/>. Acesso em: 20 fev. 2001.

[SCH 97] SCHILDT, Herbert. **Borland C++:** Completo e Total. São Paulo: Makron Books do Brasil, 1997. 1114p.

[SEI 2001] SEIKO INSTRUMENTS INC. **Hardware Specification S7600A TCP/IP Protocol Network LSI.** 2001. Disponível em: <http://www.seiko-usa-ecd.com/>. Acesso em: 20 fev. 2001.

[SHA 95] SHANLEY, Tom; ANDERSON, Don. **ISA System Architecture.** Reading: Addison Wesley Longman, 2000. 517p.

[SPU 2002] SPURGEON, Charles. **Quick Reference Guide to 10 Mbps Ethernet.** 2002. Dispoível em: <http://ethermanage.com/ethernet>. Acesso em: 17 abr. 2002.

[TAN 97] TANENBAUM, Andrew. **Redes de Computadores.** Rio de Janeiro: Campus, 1997. 990p. p.50 – 75.

[VAR 98] VARHOL, Peter. Designing Communications for Web Devices. **Computer Design,** Northbrook, v.37, n.9, p. 49-57, Sept. 1998.