Pontifícia Universidade Católica do Rio Grande do Sul

Faculdade de Engenharia – Faculdade de Informática

Curso de Engenharia de Computação

PrimeSec: Sistema de Detecção de Intrusão em Redes de Computadores sobre uma Plataforma Multiprocessada

Volume Final de Trabalho de Conclusão

Autores
Guilherme Montez Guindani
Hugo Artur Weber Schmitt

Orientador
Prof. Fernando Gehm Moraes

Porto Alegre, 30 de novembro de 2006.

Índice

1 Introdução 1
1.1 Motivação 1
1.2 Objetivos 2
1.3 Estrutura deste Documento 2
2 Referencial Teórico 3
2.1 O Modelo OSI-RM 3
2.2 A Definição dos Pacotes Ethernet 4
2.3 A Definição dos Datagramas IP 6
2.4 A Definição dos Segmentos TCP 8
2.5 A Definição dos Datagramas UDP 12
2.6 A Definição das Mensagens ICMP 12
3 NIDS – Sistemas de Detecção de Intrusão em Redes 15
3.1 Snort – Descrição de operação 15
3.2 Snort – O formato das regras 18
4 Implementação de NIDS em Hardware 19
4.1 SPP-NIDS – Um NIDS em Hardware 19
4.1.1 Visão Geral da arquitetura do SPP-NIDS 19
4.1.2 Estratégia de prototipação da SPP-NIDS 20
4.1.3 A estrutura do protótipo SPP-NIDS 20
4.1.4 A pico-CPU 21
4.1.5 O cluster GIOIA 24
4.1.6 O wrapper do cluster GIOIA 26
4.1.7 O módulo serial 26
4.1.8 O processador MR2 26
4.2 FORCE10 P-Series 27
5 PrimeSec - Um NIDS em Hardware Multiprocessado 29
5.1 Visão geral da arquitetura do PrimeSec 29
5.2 O módulo MAC Ethernet 10/100 Mbps 30
5.2.1 Visão geral da arquitetura do módulo MAC Ethernet 30
5.2.2 Descrição das interfaces externas do módulo MAC Ethernet 30
5.3 O Processador Plasma 32
5.4 O cluster GIOIA 33
5.5 Visão geral das interfaces internas do PrimeSec 33
5.5.1 A interface de recepção de pacotes recebidos da rede sem segurança 33
5.5.2 A interface de verificação de pacotes 35
5.5.3 A interface de transmissão de pacotes à rede segura 35
5.5.4 A interface de recepção de pacotes da rede segura 36
5.5.5 A interface de configuração e monitoração do sistema 37
5.5.6 A interface de transmissão de pacotes à rede sem segurança 39
6 Estratégia de Validação Funcional do PrimeSec 40
6.1 Estrutura do testbench utilizada 40
6.2 Validação da interface de recepção de pacotes da rede sem segurança 40
6.3 Validação da interface de verificação dos pacotes 42
6.4 Validação da interface de transmissão de pacotes à rede segura 43
6.5 Validação da interface de recepção de pacotes da rede segura 44
6.6 Validação da interface de configuração e monitoração do sistema 45
6.7 Validação da interface de transmissão de pacotes a rede sem segurança 46
7 Conclusões e Trabalhos Futuros 48
8 Referências Bibliográficas 49

Índice de Figuras

Figura 1 - O OSI-RM e um exemplo típico de sistema operacional de rede [TOR01]. 4
Figura 2 - Quadro Ethernet (IEEE 802.3 básico) e Ethernet II (DIX) [TOR01]. 5
Figura 3 - Relação dos protocolos da pilha TCP/IP com o modelo OSI [TOR01]. 7
Figura 4 - Estrutura do cabeçalho de um datagrama IP. 7
Figura 5 - Encapsulamento dos dados e dos diversos protocolos na pilha TCP/IP [TOR01]. 10
Figura 6 - Um exemplo que ilustra a rede na visão do protocolo TCP. 10
Figura 7 - O handshake de três vias usado no fechamento de uma conexão. 11
Figura 8 - Estrutura do cabeçalho de um segmento TCP. 11
Figura 9 - Formato do cabeçalho de um datagrama UDP. 12
Figura 10 - Formato do cabeçalho de uma mensagem ICMP. 14
Figura 11 - Árvore de diretórios de instalação do Snort e trecho do diretório das regras. 17
Figura 12 - Arquitetura simples com NIDS e roteador opcional. 17
Figura 13 - Rede com DMZ Firewall e Snort. 18
Figura 14 – Diagrama de blocos da arquitetura do SPP-NIDS. 19
Figura 15 – Estrutura geral do protótipo da plataforma SPP-NIDS. 21
Figura 16 - Organização das memórias DPRAMs utilizadas pelas pico-CPUs. 22
Figura 17 - Conjunto de sinais de entrada/saída de dados e controle de uma pico-CPU. 22
Figura 18 – Estrutura interna do bloco de dados de uma pico-CPU. 24
Figura 19 – Interface interna do cluster e a interface com a CPU de controle IDS. 25
Figura 20 – Interfaces e estrutura parcial do wrapper do cluster. 26
Figura 21 - Modelo de filtros usado na plataforma P-Series. 27
Figura 22 - Firewall P-Series, da empresa Force10. 28
Figura 23 – Visão geral da arquitetura da plataforma PrimeSec. 29
Figura 24 - Visão geral da arquitetura do MAC Ethernet e seu contexto de uso. 30
Figura 25 – Interface com a camada física Ethernet PHY e a interface com os módulos internos do sistema. 31
Figura 26 - Visão da arquitetura do processador Plasma e suas interfaces externas. 32
Figura 27 - Principais interfaces do sistema PrimeSec. 33
Figura 28 - Fluxo de dados de um pacote recebido no sistema. 34
Figura 29 – Fluxo de dados de um pacote na interface de verificação de pacotes. 35
Figura 30 - Fluxo de dados de um pacote na interface de transmissão de pacotes à rede segura. 36
Figura 31 - Fluxo de dados de um pacote na interface pacotes da rede segura. 37
Figura 32 - Fluxo de dados de um pacote na interface configuração e monitoração do sistema. 37
Figura 33 - Estrutura do cabeçalho de um pacote de configuração. 38
Figura 34 - Fluxo de dados de um pacote na interface transmissão de pacotes à rede sem segurança 39
Figura 35 - Estrutura do testbench utilizado nas simulações do sistema PrimeSec. 40
Figura 36 - Recepção de um pacote no MAC Ethernet 1. 41
Figura 37 - Operação de repasse do pacote do Buffer de Entrada ao Buffer Meio. 41
Figura 38 - Pacote sendo escrito no Buffer Meio e a interrupção gerada ao Plasma 2. 42
Figura 39 - Repasse dos bytes do conteúdo do pacote ao cluster. 42
Figura 40 - Sinalização da detecção de uma regra no pacote em comparação via interrupção. 43
Figura 41 - Operação de repasse do pacote ao Buffer de Saída para transmissão. 44
Figura 42 - Transmissão de um pacote do MAC Ethernet 2 à rede segura. 44
Figura 43 - Recepção de um pacote no MAC Ethernet 2. 45
Figura 44 - Interrupção do Plasma 3 pelo Buffer de Saída. 45
Figura 45 - Configuração das regras do cluster através da interface de configuração e monitoração do sistema. 46
Figura 46 – Troca de memória realizada pelo Plasma 3. 46
Figura 47 - Interrupção do Plasma 3 pelo Buffer de Saída 47

Índice de Tabelas

Tabela 1 – Número de regras por protocolo no Snort v.2.11. 24
Tabela 2 – Instruções do MR2, um subconjunto de instruções do MIPS 2000 [CAL04]. 27

Lista de abreviaturas

ADCCP	Advanced Data Communication Control Procedures
ANSI	American National Standards Institute
CI	Circuito Integrado
DDCMP	Digital Data Communications Message Protocol
DEC	Digital Equipment Corporation
DPRAM	Dual Port Random Access Memory
FCS	Frame Check Sequence
FPGA	Field-Programmable Gate Array
FSM	Finite State Machine
FTP	File Transfer Protocol
HDLC	High-Level Data Link Control
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronic Engineers
IP	Internet Protocol
ISO	International Standards Organization
LAN	Local Area Network
NIDS	Network Intrusion Detection System
OSI	Open Systems Interconnect
OSI-RM	Open Systems Interconnect Reference Model
RAM	Random Access Memory
SDLC	Synchronous Data Link Control
SOC	System-On-a-Chip
SPP-NIDS	Sea of Processors Platform for Network Intrusion Detection System
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VHDL	VHSIC Hardware Description Language
WAN	Wide Area Network
WWW	World Wide Web

1 Introdução

O constante crescimento das redes de computadores e sua utilização são acompanhados por um aumento das ameaças à segurança, o que vem a exigir um volume cada vez maior de processamento de dados em segurança de redes.

O maior desempenho dos processadores e o aumento da capacidade das memórias proporcionam incrementos na capacidade de processamento disponível para aplicação em segurança de rede. No entanto, existe um ponto a partir do qual a quantidade de recursos investida em hardware adicional não resultará em um aumento proporcional no desempenho dos sistemas [DES03]. Há de ser considerado que a evolução tecnológica favorece também os "intrusos", tanto mais quanto se coloca à disposição ferramentas sofisticadas de ataque disponíveis na internet [BAS04][ALL03].

Os ganhos necessários na área de segurança de redes, já demonstrados na evolução de técnicas e algoritmos, devem vir do trabalho dos desenvolvedores de *Intrusion Detection Systems* (IDS). Este foi o caso da evolução do *Snort* [CAS03], em que o desempenho do sistema aumentou várias vezes até a versão atual, devido à aplicação de algoritmos mais eficientes.

A maioria dos sistemas de computação utilizados atualmente são sistemas mono processados com um único fluxo de execução seqüencial, o que os limita em desempenho. Uma alternativa a esses sistemas são máquinas paralelas. Entretanto, estas máquinas têm um custo muito mais elevado, em razão da especialização necessária tanto da eletrônica quanto do software empregado. Sistemas operacionais com capacidade de processamento paralelo, linguagens de programação e algoritmos paralelos são específicos por aplicação e exigem práticas diferenciadas em sua produção, aplicação e manutenção e são de baixo desempenho se desenvolvidos para uso geral.

Uma solução para equacionar custo e desempenho é oferecida quando temos co-processadores de uso específico [HAR97][HAU97], em que as unidades especializadas de processamento paralelo integram-se aos sistemas tradicionais com vantagem no custo global do sistema.

A tecnologia de circuitos integrados (CIs) de lógica reconfigurável, como por exemplo FPGAs, permite liberdade na concepção e prototipação de arquiteturas inovadoras. Estes componentes, baseados em um arranjo bidimensional que repete uma mesma célula básica de lógica configurável, possuem um paralelismo nativo. Além deste arranjo básico, outras estruturas mais complexas como blocos de memória ou mesmo processadores são integrados ao componente, aumentando consideravelmente seu poder de computação. Componentes fabricados com esta tecnologia, e com grande densidade de elementos construtivos, tornaram possível a implementação em FPGAs de sistemas computacionais completos em um único CI, conhecidos por SoCs (*System-on-a-Chip*) [BER01][GUP97].

1.1 Motivação

A necessidade de assegurar o funcionamento de sistemas de computação tem exigido esforços crescentes no sentido de evitar o que se convencionou chamar de intrusão, ou seja, o uso não autorizado ou o uso malicioso de um sistema de computação.

Isto levou à criação de sistemas de detecção de intrusão ou IDS (Intrusion Detection System).

No contexto das redes de computadores, foram criados os NIDS (Network Intrusion Detection System), IDSs especializados na vigilância sobre o tráfego de redes, como é o caso do programa Snort [CAS03][NOR03][NOR03a], aplicação de fonte aberto amplamente utilizada em todo mundo com esta finalidade.

O desenvolvimento de arquiteturas paralelas, integradas em um único circuito integrado abre novas perspectivas relacionadas ao desempenho e a segurança no tratamento de intrusões. Estas arquiteturas ainda são raras no mercado. Um exemplo é a plataforma P-Series da empresa Force10 [FOR06].

1.2 Objetivos

Este trabalho estende o trabalho de mestrado de Luís Carlos Mieres Caruso [CAR05a], o qual propôs uma arquitetura paralela e implementou um primeiro protótipo deste, que é voltada para detecção das regras do Snort. O presente trabalho completa esta arquitetura, integrando-a a interfaces de rede e processadores.

O objetivo deste trabalho é aplicar os conhecimentos adquiridos durante o curso, desenvolvendo uma estrutura de suporte a um co-processador de detecção de padrões. Fazendo isto, tem-se um protótipo de um sistema de NIDS em hardware, denominado Primesec. O PrimeSec utilizará este co-processador para o processamento de padrões, analisando o tráfego entre redes de computadores. Este trabalho visa tornar o conjunto desenvolvido em um pré-produto, ou seja, operacional ao ponto de ser possível realizar testes do protótipo em campo.

1.3 Estrutura deste Documento

O presente documento é organizado como segue. O Capítulo 2 apresenta conceitos de redes, protocolos e pacotes, necessários para a compreensão do presente trabalho. O Capítulo 3 conceitua o que são os NIDS (sistemas de detecção de intrusão em redes), apresentando o Snort, um NIDS em software. O Capítulo 4 apresenta duas implementações de NIDS em hardware: (i) o SPP-NIDS, utilizado como base para o desenvolvimento do presente Trabalho de Conclusão; (ii) um exemplo de arquitetura comercial com características similares à arquitetura proposta neste trabalho. O Capítulo 5 apresenta a principal contribuição do TC: o detalhamento da arquitetura e da implementação do sistema PrimeSec. O Capítulo 6 apresentação a validação da arquitetura PrimeSec. O Capítulo 7 apresenta as conclusões do trabalho e sugere futuros temas de pesquisa em arquiteturas para detecção de intrusão em redes.

2 Referencial Teórico

Neste Capítulo apresentam-se os conceitos básicos de redes relacionados ao tema do presente Trabalho de Conclusão: a definição do modelo OSI-RM, a definição de pacotes Ethernet, IP, TCP, UDP e ICMP.

2.1 O Modelo OSI-RM

O Modelo de Referência para Interconexão de Sistemas Abertos (em inglês, *Open Systems Interconnection Reference Model* ou OSI-RM), proposto pela *International Standards Organization* (ISO), permite a estratificação, o projeto e a implementação dos protocolos padronizados em redes de comunicação. O OSI-RM divide o tratamento de informações de rede em sete níveis de abstração. Estes níveis estão numerados e definidos a seguir:

- **Nível 1 (Físico)**: define o meio de transmissão, suas características elétricas e mecânicas. Exemplos de normas aplicadas neste nível são: RS-232, RS-422A, RS-485 da EIA (*Electronic Industries Association*), etc;
- **Nível 2 (Enlace de Dados)**: estabelece, mantém e permite acesso ao meio físico, empacota mensagens para transmissão e verifica a integridade das mensagens recebidas. Exemplo de normas e padrões aplicados neste nível são: HDLC(ISO), ADCCP(ANSI), DDCMP(DEC), SDLC e BSC (IBM);
- **Nível 3 (Rede)**: controla o encaminhamento das mensagens, gerenciando o seu fluxo dentro e entre redes. Exemplos de normas aplicadas a este nível são: X.20, X.21, X.25 da CCITT (*International Consulting Committee on Telephone and Telegraph*);
- **Nível 4 (Transporte)**: cria e mantém a conexão entre estações, permitindo a conexão lógica segura entre estações;
- **Nível 5 (Sessão)**: estabelece e controla as sessões (mensagens estruturadas) entre duas estações da rede.
- **Nível 6 (Apresentação)**: proporciona uma representação e formato geral de informação trocada entre duas estações da rede. Um exemplo neste nível é a codificação MIME.
- **Nível 7 (Aplicação)**: define regras de sintaxe, semântica e gramáticas para troca de informações no nível de transporte de arquivos, acesso a bases de dados e uso compartilhado de recursos entre duas estações da rede.

A hierarquia proposta pelo OSI-RM é apresentada no lado esquerdo da Figura 1. Na direita da figura, comparativamente, observa-se uma equivalência com um sistema operacional de rede típico que implementa cada nível do modelo OSI-RM [TOR01].

Em geral, para que um módulo processador possa operar em uma rede de computadores, devem ser instalados recursos de hardware e software que complementem seus dispositivos e seu sistema operacional local. Em uma rede local, o hardware adicional necessário se constitui, tipicamente, de uma placa de interface de rede e um software, que definimos como *sistema operacional de rede*. Os componentes típicos de um sistema operacional de rede são mostrados na Figura 1 à direita, comparados com o OSI-RM. Tal sistema engloba os seguintes componentes:

- As aplicações cliente-servidor de uso geral;

- Um conjunto de módulos implementando os protocolos;
- Um ou mais módulos de software básico (*drivers*) com implementações de protocolos de comunicação.

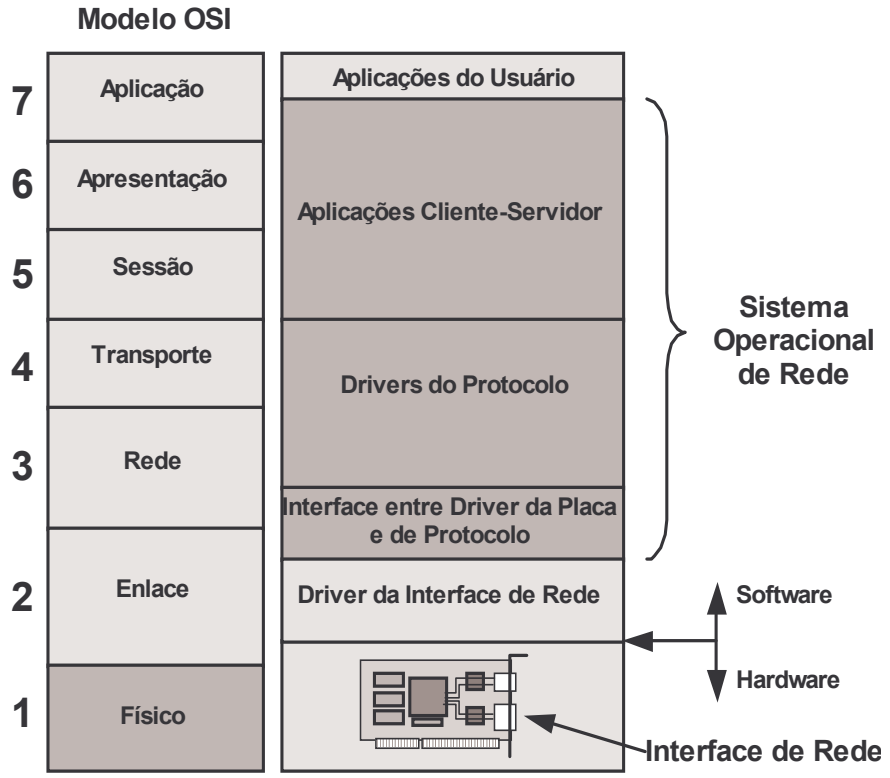


Figura 1 - O OSI-RM e um exemplo típico de sistema operacional de rede [TOR01].

O controle e configuração da interface de rede são realizados pelo *driver* da interface de rede, que se comunica com o sistema operacional de rede através de uma interface padronizada.

2.2 A Definição dos Pacotes Ethernet

Os termos *pacote*, *quadro* e respectivas siglas serão aqui definidos, procurando manter coerência com o padrão Ethernet IEEE 802.3 [ETH06], pois serão utilizadas constantemente nesta Seção e nas outras que se seguem. Conforme representado na Figura 2, o termo *pacote* (em inglês, *packet*) refere-se à um bloco de dados existente em qualquer nível de abstração, e o termo *quadro* (em inglês, *frame*), quadro de dados ou mensagem, se refere à porção do pacote que é transferida de ou para o nível superior ao subnível MAC.

A Figura 2 detalha os dois tipos de quadros usados na transmissão de dados via tecnologia Ethernet. Há uma diferença sutíl entre o padrão Ethernet II (DIX) anterior à padronização pela IEEE e a norma IEEE 802.3 na estrutura dos seus quadros. No quadro Ethernet II há um campo chamado tipo (*type*), porém no padrão IEEE 802.3 o campo correspondente contém a informação do tamanho do quadro (*length ou LE*). Esta diferença, contudo não causa problemas para implementar o método de transmissão no meio físico, o CSMA/CD (do inglês, *Carrier Sense Multiple Access with Collision Detection*) padrão IEEE 802.3, devido à codificação empregada. A interpretação dos bytes *LE* ou *type* é a seguinte:

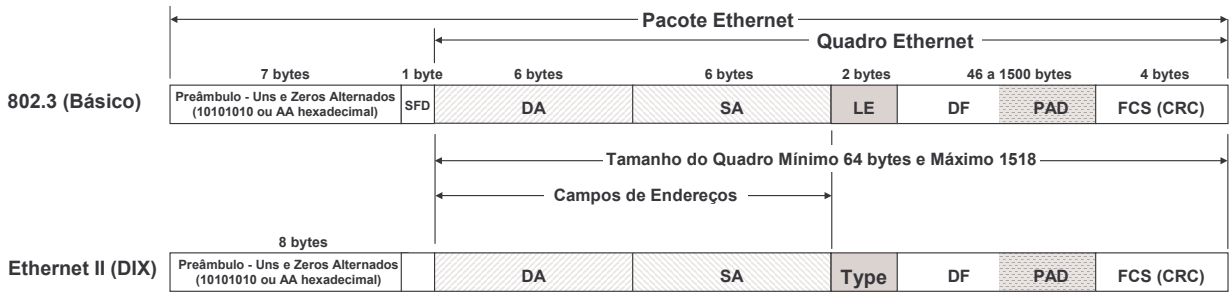


Figura 2 - Quadro Ethernet (IEEE 802.3 básico) e Ethernet II (DIX) [TOR01].

- Se o valor é menor ou igual ao tamanho máximo dos dados (1500 bytes), então o campo é interpretado como tamanho de quadro e representa a quantidade de bytes de dados contidos no campo seguinte, o campo de dados (*data field, DF*).
- Valores no intervalo entre 1501 e 1535 são considerados impróprios, pois o pacote IEEE 802.3 Básico tem tamanho máximo de 1526 somando-se o conjunto de bytes (1500) e mais as informações de sincronização, cabeçalho e controle de erros. Uma margem de segurança de dez bytes foi inserida para determinar uma diferenciação no campo LE/Type.
- Se o valor é maior ou igual a 1536 bytes (0600 hexadecimal), então o campo indica a natureza do protocolo utilizado e o campo é interpretado como de tipo.

Assim, as interpretações de tamanho e tipo são mutuamente exclusivas. Quando é utilizada a codificação de tipo. Executar a correta operação para o tamanho do quadro é responsabilidade da camada superior ao subnível MAC.

A Figura 2 ilustra os sete campos de um quadro 802.3 (básico), comparado com um quadro Ethernet II DIX, um padrão industrial *de facto* que precedeu a norma IEEE 802.3 [SPU00]. As informações contidas em cada um dos campos do quadro padrão 802.3 são as seguintes:

- **Preâmbulo:** Indica o início da transmissão do quadro. Este campo de 7 bytes é usado para permitir ao circuito de sinalização física (em inglês, *Physical Signaling* ou PLS) sincronizar-se com o quadro recebido. Cada byte deste campo é formado pela seqüência 10101010;
- **Delimitador de início de quadro (em inglês, *Start Frame Delimiter* ou *SFD*):** O delimitador é a seqüência binária 10101011 e indica o começo de um quadro imediatamente após os dois últimos bits (11);
- **Endereço destino:** O endereço da estação ou estações, a quem o quadro é destinado (em inglês, *Destination Address* ou DA) e seu tamanho é de 48 bits (6 bytes);
- **Endereço fonte:** Este campo indica o endereço da estação (em inglês, *Source Address* ou SA) que iniciou a transmissão. Este campo possui tamanho de 48 bits (6 bytes);
- **Tamanho do quadro:** Este campo, de 16 bits (2 bytes), indica o tamanho do quadro (em inglês, *length* ou LE), exceto os campos Preâmbulo e SFD [SPU00]. Para o padrão Ethernet II DIX, este campo é denominado tipo (em inglês, *type*). O campo de tipo contém um código específico de protocolo do quadro;
- **Dados (em inglês, *Data Field* ou DF):** Possui um tamanho mínimo de 46 bytes e máximo de 1.500 bytes. Este campo contém os dados a enviar através da rede. Se a quantidade de dados a ser enviada for menos de 46 bytes, então um conjunto de bits de preenchimento (denominados em inglês de *padding* bits ou PAD) é acrescentado ao final dos dados para completar o tamanho

mínimo;

- **Seqüência de verificação de quadro:** O *Frame Check Sequence* (FCS) implementa um código de verificação de erro do tipo *Cyclic Redundancy Check* (CRC). Este valor é computado como uma função dos conteúdos dos campos endereço destino (DA), endereço fonte (SA), tamanho do quadro (LE) e dados (DF+PAD, quando este último existir). O FCS possui 32 bits (4 bytes) de comprimento.

2.3 A Definição dos Datagramas IP

Uma rede de computadores deve oferecer um sistema de comunicação confiável e transparente. Para isto foi criado um serviço que esconde os limites da camada física de rede, provendo os benefícios de redes virtuais. As redes virtuais são uma abstração utilizada pelos desenvolvedores, permitindo a estes uma total liberdade na escolha de endereços, formato dos pacotes e aplicação de técnicas, independentemente das limitações da rede física.

O endereçamento é um componente crítico na abstração de redes virtuais. Para dar a impressão de uma rede única e uniforme, todos os computadores hospedeiros (do inglês, *host*) devem usar um esquema de endereçamento uniforme. Infelizmente, os endereços físicos não são suficientes, pois uma rede virtual pode englobar múltiplas tecnologias de rede física, cada uma com seu próprio formato de endereçamento. Desta forma, os endereços usados por duas tecnologias podem ser incompatíveis devido aos diferentes tamanhos ou formatos.

Para garantir a uniformidade no endereçamento para todos os hospedeiros, os protocolos da camada de rede definem um esquema de endereçamento independente dos endereços físicos dos dispositivos que estão se comunicando. Mesmo que o esquema de endereçamento na rede virtual seja uma abstração criada por software, os endereços do protocolo são usados como destino e origem da mesma forma que os endereços em uma rede física. Para enviar um pacote através da rede virtual, o transmissor coloca o endereço de destino no pacote e o envia ao software de protocolo para transmissão. Cada computador hospedeiro conhece apenas os endereços virtuais dos outros hospedeiros, comunicando-se através destes endereços sem ter conhecimento da rede física que liga cada um deles.

Na pilha de protocolos TCP/IP o esquema de endereçamento é realizado pelo Protocolo Internet (do inglês, *Internet Protocol* – IP) , o protocolo de rede virtual mais usado no mundo e base da Internet. O padrão IP especifica que cada hospedeiro tenha um número de 32 bits diferente, conhecido como o Endereço do Protocolo Internet (do inglês, *Internet Protocol Address*), que é comumente abreviado para Endereço IP. Em relação ao modelo OSI-RM o protocolo IP faz, em parte, o papel do nível 3 (Rede) como mostra a Figura 3.

O protocolo IP utiliza uma estrutura de dados conhecida como Datagrama IP (do inglês, *IP Datagram*) para enviar o pacote através da rede virtual. Esta estrutura é formada por um cabeçalho (do inglês, *header*) que contém as informações de controle de para onde e como o pacote vai ser transmitido e um campo de dados ou carga útil (do inglês, *payload*).

A quantidade de dados presente em um datagrama IP não é fixa. O transmissor escolhe uma quantidade de dados apropriada à cada aplicação de camadas superiores. O datagrama IP ainda deverá ser transportado na rede por um protocolo inferior a ele. O caso mais comum em redes computadores é este protocolo o Ethernet. Isso limita a capacidade de dados a serem transmitidos em um pacote IP. No protocolo IP esta limitação é contornada com o uso de fragmentação do

datagrama IP. Quando o tamanho de um datagrama IP excede a capacidade de um quadro da camada inferior, o protocolo IP fragmenta o datagrama maior em partes menores e os transmite como vários datagramas menores. A Figura 4 exibe a estrutura de um datagrama IP mostrando a estrutura do cabeçalho deste.

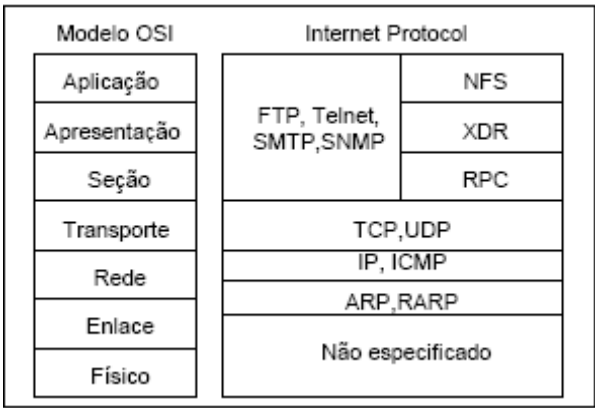


Figura 3 - Relação dos protocolos da pilha TCP/IP com o modelo OSI [TOR01].

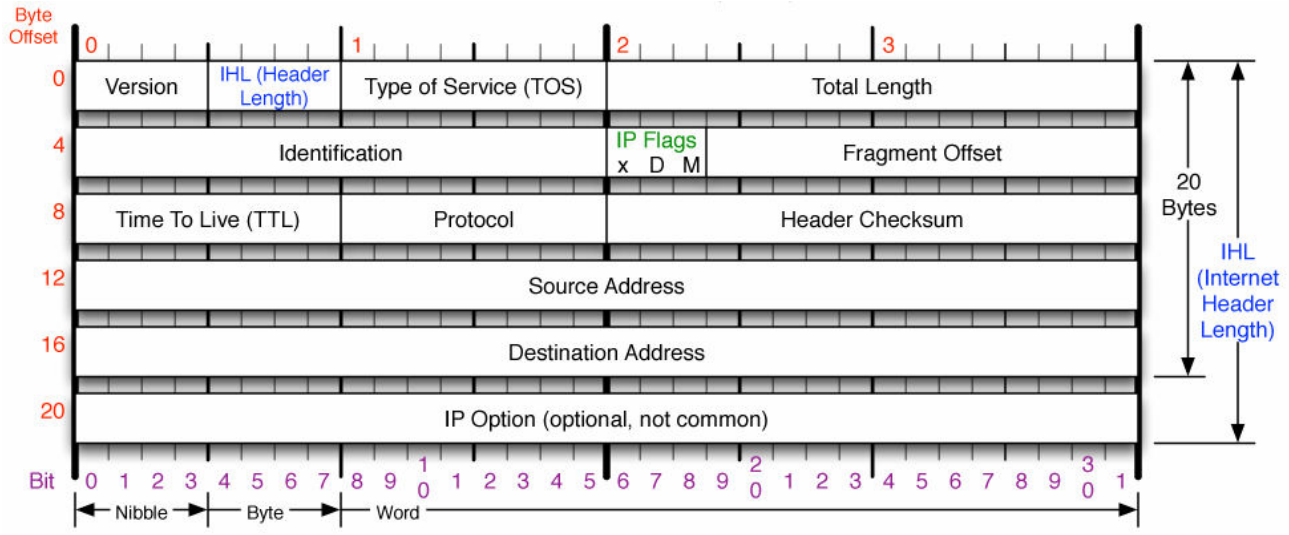


Figura 4 - Estrutura do cabeçalho de um datagrama IP.

O cabeçalho contém informações de roteamento e controle do datagrama, e possui os seguintes campos:

- **Versão (*Version*):** Este campo identifica a versão do protocolo IP usada no datagrama. Datagramas recebidos com versões não conhecidas devem ser simplesmente ignorados. A versão mais utilizada é a versão 4, mas existe uma versão mais nova, a versão 6, que é utilizada somente em algumas redes locais e nos roteadores principais da internet;
- **Tamanho do Header (*Header Length*):** Este campo define o comprimento do cabeçalho IP em múltiplos de 4 bytes. O tamanho variável do cabeçalho permite o uso de campos opcionais, mas dificulta o processamento. Desta forma, o uso de cabeçalhos de tamanho não padrão deve ser evitado;
- **Tipo de Serviço (*Type of Service*):** Este campo define a qualidade do serviço (QOS) desejada, mas é ignorado na maioria das implementações. Com o uso deste campo é possível escolher as prioridades de serviço para aplicações do tipo interativas, de transferência em massa ou de

tempo real;

- **Tamanho Total (*Total Length*):** Este campo define o tamanho total, em bytes, do datagrama (ou de um fragmento), incluindo o cabeçalho. Todos os hospedeiros devem aceitar datagramas de, pelo menos, 576 bytes, o valor máximo a ser usado em uma sub-rede pode ser negociado entre os hospedeiros;
- **Identificador (*Identification*):** Este campo define um número único para um datagrama IP que será fragmentado. Cada fragmento do datagrama terá o mesmo identificador do primeiro fragmento;
- ***Flags*:** *Flags* para a resolução de pacotes fragmentados. Este campo, de 3 bits possui 2 bits relevantes:
    - **Não Fragmentar *(Do not Fragment)*** - que informa que o datagrama não deve ser fragmentado.
    - **Mais Fragmentos *(More Fragments)*** - que indica a existência de mais fragmentos após a posição deste. Ou seja, se este bit não estiver habilitado, este é o último fragmento da cadeia;
- ***Offset* de Fragmento (*Fragment Offset*):** Este campo determina a posição relativa deste fragmento em relação ao datagrama original. Como o campo *Flags* consome 3 bits (um é reservado), este campo é utilizado sempre em múltiplos de 8;
- **Tempo de Vida (*Time to Live*):** A cada nó (roteador) por onde o datagrama passa, este campo deve ser decrementado de uma unidade. Quando o valor do TTL chega a zero, o datagrama é descartado e uma mensagem enviada ao emissor, através do protocolo ICMP. O objetivo de toda esta operação é evitar que erros nas tabelas de roteamento permitam que alguns datagramas fiquem circulando indefinidamente pela rede;
- **Protocolo (*Protocol*):** Este campo é o identificador do protocolo usado na camada imediatamente acima do IP. Os protocolos mais usados na internet são: TCP, UDP e ICMP;
- ***Checksum* do Cabeçalho (*Header Checksum*):** Este campo garante a integridade do cabeçalho IP, mas não garante a integridade dos dados. Caso haja erro neste, o datagrama deve ser descartado;
- **Endereço IP de Origem (*Source IP Address*):** Este campo informa o endereço IP da origem do datagrama;
- **Endereço IP de Destino (*Destination IP Address*):** Este campo informa o endereço IP do destino do datagrama;
- **Opções do IP (*IP Options*):** Este campo permite o uso de serviços extras do protocolo IP. Todas as implementações de IP devem suportar todas as opções válidas;

2.4 A Definição dos Segmentos TCP

Esta Seção examina o protocolo TCP, o protocolo de transporte mais utilizado na pilha de protocolos TCP/IP.

O protocolo TCP utiliza o serviço de datagramas oferecido pelo IP para enviar dados para outro computador, e provê um serviço de entrega de dados confiável para os programas aplicativos.

O TCP deve compensar a perda ou atraso de pacotes ocorridos na rede para garantir uma transmissão eficiente de dados, e precisa fazer isso sem sobrecarregar os protocolos de camadas inferiores ou os roteadores da rede.

Confiança na transmissão é responsabilidade de um protocolo de transporte; aplicações interagem com o serviço de transporte para enviar e receber dados. Na pilha de protocolos TCP/IP, o Protocolo de Controle de Transmissão (do inglês, *Transmission Control Protocol* ou TCP) provê um serviço de transporte confiável. Conseqüentemente, a maioria das aplicações de rede utiliza o TCP.

Do ponto de vista de um programa aplicativo, o serviço oferecido pelo TCP tem sete características principais [COM97]:

- Orientação a Conexões: O TCP provê um serviço orientado a conexão, no qual uma aplicação deve primeiramente requisitar uma conexão com o destino, e só então usar esta conexão para transferir dados;
- Comunicação Ponto-à-Ponto: Cada conexão TCP tem exatamente duas extremidades;
- Completa Confiança: O TCP garante que todos os dados que são enviados pela rede serão entregues exatamente como foram transmitidos, sem nenhum dado faltando ou fora de ordem;
- Comunicação *Full Duplex*: Uma conexão TCP permite que os dados fluam nas duas direções, e permite que o programa aplicativo envie dados a qualquer momento. O TCP pode memorizar dados de entrada e saída nos dois sentidos, sendo possível que a aplicação envie dados e em seguida continue o processamento enquanto os dados estão sendo transmitidos;
- Interface Contínua: Dize-se que o TCP provê uma interface contínua, na qual uma aplicação envia uma seqüência contínua de bytes através da conexão. Isto é, o TCP não mantém um registro do que já foi transmitido, e não garante que uma estrutura de dados criada por uma aplicação origem seja entregue em partes de mesmo tamanho que a original na aplicação destino, onde esta estrutura pode ser fragmentada e deverá ser remontada antes de chegar à aplicação destino;
- Início de Conexão Segura: O TCP exige que quando duas aplicações criam uma conexão, ambas devem concordar com esta nova conexão. Pacotes duplicados usados em conexões anteriores não serão considerados respostas válidas, desta forma não irão interferir na nova conexão;
- Fechamento de Conexão Confiável: Um programa aplicativo pode abrir uma conexão, enviar uma quantidade de dados arbitrária, e depois requisitar o fechamento da conexão. O TCP garante a entrega de todos os dados antes de fechar a conexão.

O TCP utiliza o IP para enviar mensagens. Cada mensagem TCP é encapsulada em um datagrama IP, este por sua vez é encapsulado em um, ou mais frames Ethernet que são então transmitidos pela rede. Quando um datagrama chega ao seu destino, o IP passa o seu conteúdo ao TCP. Note que apesar de o TCP usar o IP para enviar mensagens, o IP não lê ou interpreta estas mensagens. Deste modo, o TCP trata o IP como um sistema de pacotes de comunicação, e o IP trata cada mensagem TCP como dados a serem transmitidos. A Figura 5 mostra os diferentes níveis de encapsulamento na transmissão de dados pela rede.

A Figura 6 mostra um exemplo de rede que contém dois hospedeiros e um roteador e ilustra a relação entre os softwares de TCP e IP.

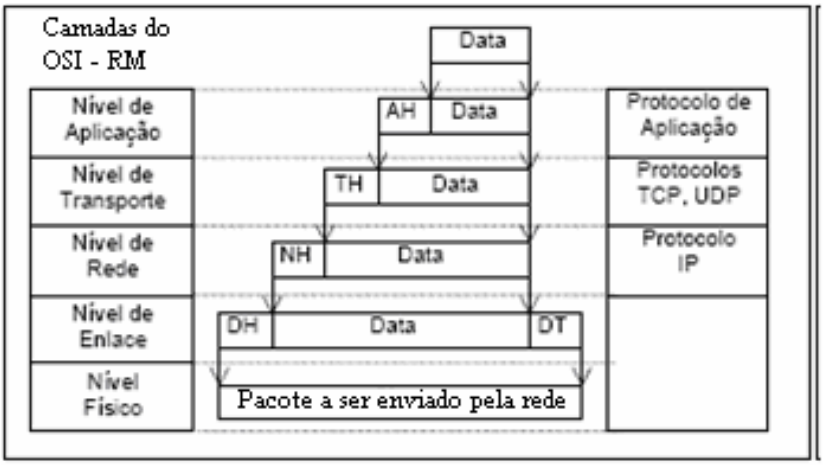


Figura 5 - Encapsulamento dos dados e dos diversos protocolos na pilha TCP/IP [TOR01].

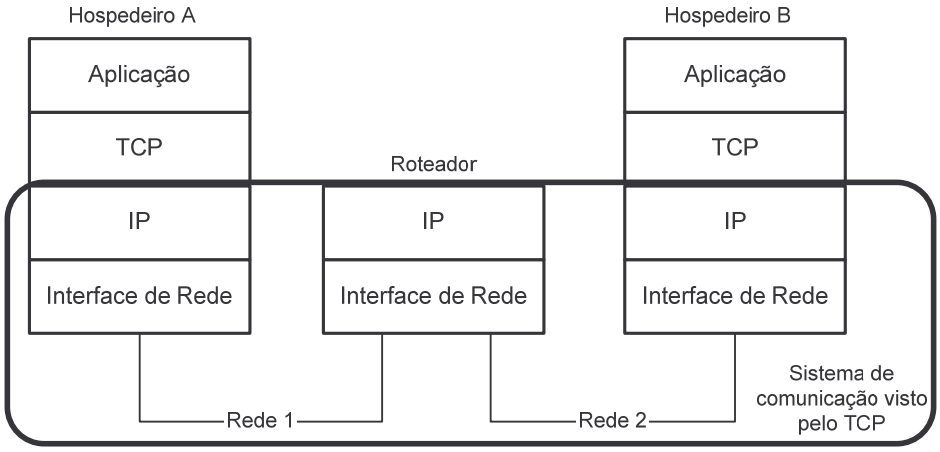


Figura 6 - Um exemplo que ilustra a rede na visão do protocolo TCP.

Como mostra a figura acima, o TCP é necessário em cada ponta da conexão virtual de rede, mas não nos roteadores intermediários. Sob o ponto de vista do TCP, toda a rede é um sistema de comunicação que aceita e entrega mensagens sem mudar ou interpretar o conteúdo destas.

Para garantir que as conexões são estabelecidas e terminadas de forma confiável, o TCP utiliza um handshake de três vias (do inglês, 3-way Handshake) na qual três mensagens são trocadas entre as partes. O TCP usa o termo segmento de sincronização (do inglês, synchronization segment ou SYN segment) para descrever as mensagens no sistema de handshake de três vias usadas na criação de conexões, e o termo segmento de término (do inglês, finish segment ou FIN segment) para descrever as mensagens usadas para fechar uma conexão. A Figura 7 ilustra o handshake de três vias usado para fechar uma conexão, mesmo procedimento utilizado no estabelecimento de conexão.

O TCP utiliza um único formato para todas as mensagens, incluindo as mensagens que contêm dados válidos, confirmações, e mensagens que fazem parte do handshake de três vias usado para criar ou terminar uma conexão. O TCP utiliza o termo segmento (do inglês, *segment*) ao se referir a esta mensagem. A Figura 8 ilustra o formato de um segmento.

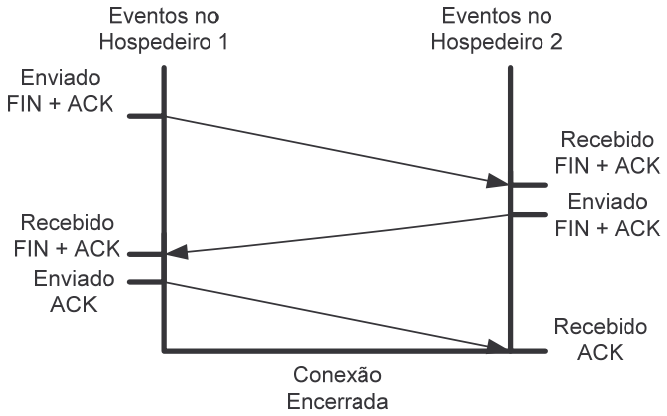


Figura 7 - O handshake de três vias usado no fechamento de uma conexão.

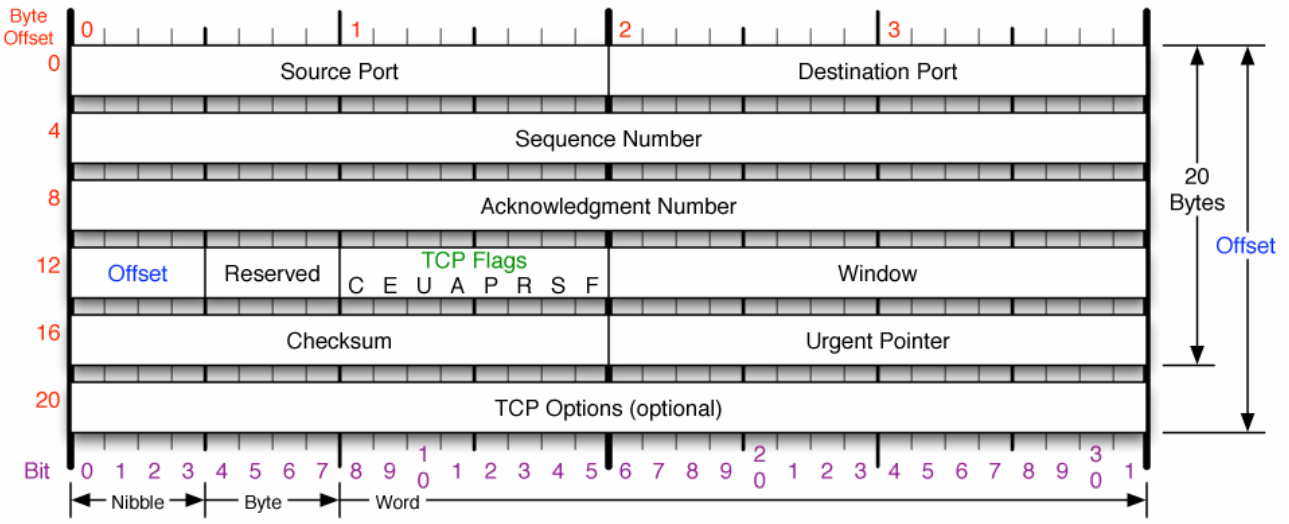


Figura 8 - Estrutura do cabeçalho de um segmento TCP.

O cabeçalho de um segmento TCP possui os seguintes campos:

- **Porta Origem (*Source Port*):** Identifica a aplicação de origem;
- **Porta Destino (*Destination Port*):** Identifica a aplicação de destino;
- **Número de Seqüência (*Sequence Number*):** Identifica a posição deste segmento no fluxo de dados e cada conexão possui um fluxo de dados particular;
- **Número de Confirmação (*Acknowledgement Number*):** Utilizado para confirmar o recebimento de segmentos enviados anteriormente e especifica o próximo segmento aguardado;
- ***Offset*:** Define o tamanho do cabeçalho em palavras de 32 bits;
- **Reservado (*Reserved*):** Reservado para testes e novas implementações do TCP;
- ***Flags*:** Possui as informações de conexão do TCP;
- **Janela (*Window*):** Indica o tamanho (disponível) do buffer do receptor e é usado pelo receptor para indicar ao transmissor uma diminuição no fluxo de dados transmitidos;
- ***Checksum*:** Contém informações usadas na verificação de erros do protocolo;
- **Ponteiro de Urgência (*Urgent Pointer*):** Usado pela origem para indicar onde se encontra algum dado urgente dentro do segmento;
- **Opções (*Options*):** Usado para a configuração de opções, se houver;

2.5 A Definição dos Datagramas UDP

O Protocolo de Datagramas de Usuário (do inglês, *User Datagram Protocol,* ou *UDP*) pode ser descrito de forma análoga ao TCP. O UDP provê às aplicações acesso direto ao serviço de entrega de datagramas, como o serviço de entrega que o IP provê. O UDP é pouco confiável, sendo um protocolo não orientado a conexão. O termo "pouco confiável" significa que não há técnicas no protocolo para confirmar que os dados chegaram ao destino corretamente.

O UDP faz a entrega de mensagens independentes, designadas por datagramas, entre aplicações ou processos, em sistemas hospedeiros. A entrega não é confiável, porque os datagramas podem ser entregues fora de ordem ou até descartados. O formato de um datagrama UDP é bem simples, ele contém um pequeno cabeçalho e um campo de dados. A Figura 9 ilustra o formato de um datagrama UDP.

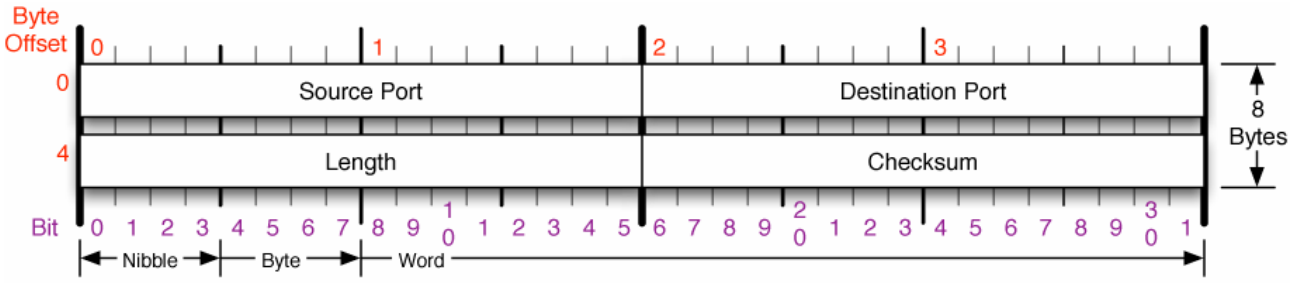


Figura 9 - Formato do cabeçalho de um datagrama UDP.

O cabeçalho UDP possui os seguintes campos:

- **Porta Origem (*Source Port*):** Identifica a aplicação de origem;
- **Porta Destino (*Destination Port*):** Identifica a aplicação de destino;
- **Comprimento (*Length*):** Contém um contador de bytes no datagrama UDP. O valor mínimo é oito, sendo este só o comprimento do cabeçalho;
- ***Checksum*:** Contém informações usadas na verificação de erros do protocolo.

2.6 A Definição das Mensagens ICMP

Esta Seção examina um protocolo de relatório de erros que é integrado ao protocolo IP. São revisados os erros básicos que podem ser informados, e é explicado como e quando estas mensagens de erro são enviadas.

O serviço IP é um serviço pouco confiável, pois os datagramas podem ser descartados, duplicados, atrasados ou entregues fora de ordem durante seu trajeto entre dois hospedeiros. Desta forma, é necessário dispor de serviços e mecanismos que realizem a detecção de erros.

Problemas menos graves que a transmissão de erros resultam em condições de erro que podem ser informadas. Por exemplo, suponha que alguns caminhos físicos na rede venham a falhar, dividindo a rede em duas partes e sem nenhuma conexão entre elas. Um datagrama enviado de um hospedeiro em uma parte com destino a outra parte não poderá ser entregue.

A pilha de protocolos TCP/IP inclui um protocolo usado pelo IP para enviar mensagens de erro quando situações como a descrita acima ocorrem: o Protocolo de Controle de Mensagens da Internet (do inglês, *Internet Control Message Protocol* ou *ICMP*). Este protocolo é necessário na implementação padrão do IP. O ICMP e o IP são dois protocolos co-dependentes, o IP utiliza o ICMP quando precisa transmitir uma mensagem de erro e o ICMP utiliza o IP para transportar suas

mensagens.

O ICMP define cinco mensagens de erro e quatro mensagens informais. As cinco mensagens de erro no ICMP são:

- **Dissipador de Fonte (*Source Quench*):** Um roteador envia uma mensagem dissipadora de fonte a cada vez que este recebeu tantos datagramas que não possui mais espaço de armazenamento. Um roteador que tenha atingido seu limite de armazenamento deve descartar todos os datagramas que são recebidos. Quando descarta um datagrama, o roteador envia uma mensagem dissipadora de fonte ao hospedeiro que criou o datagrama. Quando receber uma mensagem dissipadora de fonte, o hospedeiro precisa reduzir a sua taxa de transmissão;
- **Tempo Excedido (*Time Exceeded*):** Uma mensagem de tempo excedido é enviada em dois casos:
    - toda vez que um roteador reduz o campo tempo de vida do datagrama e este chega a zero, o roteador então descarta o datagrama e envia uma mensagem de tempo excedido ao hospedeiro de origem;
    - Quando o tempo de remontagem (do inglês, *reassembly*) expirar antes que todos os fragmentos de um dado datagrama tenham sido recebidos;
- **Destino Inatingível (*Destination Unreacheable*)**: Cada vez que um roteador determinar que um datagrama não pôde ser entregue ao seu destino final, o roteador envia uma mensagem de destino inatingível ao hospedeiro que criou o datagrama. A mensagem especifica se o hospedeiro está inatingível ou se a rede na qual este está conectado está inatingível. Em outras palavras, esta mensagem de erro distingue se aquela sub-rede está desconectada do resto da rede (e.g., devido a queda de um roteador) ou o caso em que aquele hospedeiro em particular está temporariamente fora do ar (e.g., devido ao hospedeiro estar desligado).
- **Redirecionamento (*Redirect*):** Quando um hospedeiro cria um datagrama destinado à uma rede remota, este envia o datagrama a um roteador, o qual encaminha este datagrama ao seu destino. Se o roteador determina que o hospedeiro lhe enviou incorretamente um datagrama e este precisa ser enviado a um roteador diferente, o roteador que recebeu o datagrama utiliza uma mensagem de redirecionamento para que o hospedeiro corrija a rota deste. A mensagem de redirecionamento pode especificar uma mudança de um hospedeiro específico ou uma mudança de rede; a ultima opção é a mais comum;
- **Necessita Fragmentação (*Fragmentation Required*)**: Um hospedeiro ao criar um datagrama pode especificar no cabeçalho que este não pode ser fragmentado. Se um roteador verifica que este datagrama é maior que a MTU (do inglês, *Maximum Transmission Unit*) da rede em que está sendo transmitido, ele pode enviar uma mensagem Necessita Fragmentação ao transmissor, e então descarta o datagrama.

Além destas mensagens de erro, o ICMP define as seguintes mensagens informativas:

- **Pedido/Resposta de Eco (*Echo Request/Reply*):** Um pedido de eco pode ser transmitido ao software ICMP de qualquer computador. Em resposta à um pedido de eco, o software ICMP é obrigado a enviar uma mensagem de resposta de eco. A resposta leva o mesmo conjunto de dados do pedido de eco.
- **Pedido/Resposta de Máscara de Endereço (*Address Mask Request/Reply*):** Um hospedeiro, na sua inicialização, transmite a todos um pedido da máscara de endereço, e um roteador que

recebe este pedido envia uma resposta da máscara de endereço que contém a máscara de sub-rede (32-bits) em uso.

A mensagem ICMP é formada por um cabeçalho e um campo de dados. A Figura 10 ilustra o formato do cabeçalho de uma mensagem ICMP.

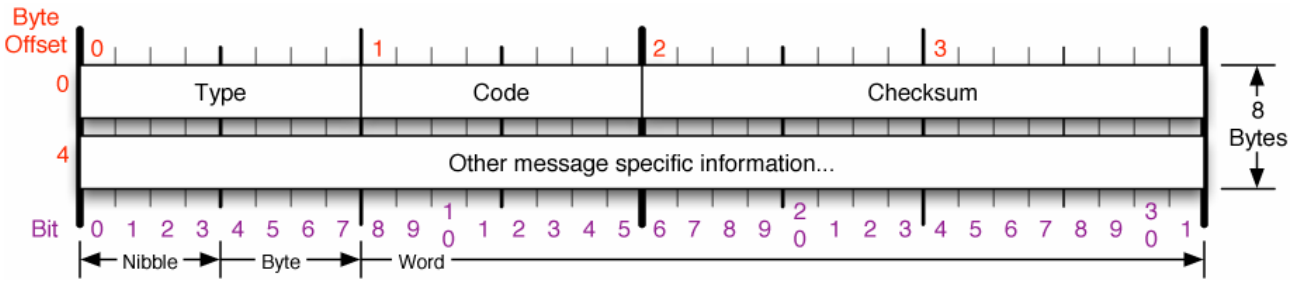


Figura 10 - Formato do cabeçalho de uma mensagem ICMP.

O cabeçalho de uma mensagem ICMP possui os seguintes campos:

- **Tipo (*Type*):** Especifica o tipo da mensagem ICMP;
- **Código (*Code*):** Qualifica a mensagem ICMP;
- ***Checksum*:** Contém informações usadas na verificação de erros do protocolo;
- **Outras informações específicas da mensagem:** Este campo pode conter mais informações da mensagem como, por exemplo, a máscara de sub-rede em uma mensagem do tipo *Address Mask Reply*.

3 NIDS – Sistemas de Detecção de Intrusão em Redes

Com o barateamento dos computadores pessoais, a interconexão entre eles, ou uma rede de computadores, tornou-se uma entidade vital no dia-a-dia. Todo tipo de informação trafega por esta rede, desde transações financeiras, informações esportivas, entretenimento, telefonia e até informações militares. Desta forma, a rede tornou-se alvo de grande cobiça por informações importantes. Esta cobiça por informação leva a rede a sofrer diversas tentativas de acesso indevidas, tentativas estas conhecidas como ataques.

Existem diversos tipos (ou técnicas) de ataques em redes de computadores. Alguns visam a extração de informação, alguns visam a paralisação de um determinado serviço ou máquina da rede, outros visam a conexão com um determinado serviço ou máquina da rede, etc. Estes ataques podem estar presentes em diferentes protocolos da rede. Existem ataques IP, ICMP, TCP, UDP entre outros.

Os ataques de redes que são focados no presente Trabalho de Conclusão são aqueles em que o código de ataque malicioso está presente no conteúdo (em inglês, *content*) do pacote nos protocolos IP, ICMP, TCP e UDP. Estes ataques em particular são detectados através de um comparador de padrões.

A necessidade de garantir o funcionamento seguro de sistemas de computação tem exigido esforços crescentes, no sentido de evitar o que se convencionou chamar de intrusão, ou seja, o uso não autorizado ou o mal uso de um sistema de computação [PTA03][KEM02]. Isto levou à criação de sistemas de detecção de intrusão ou IDSs (*Intrusion Detection Systems*), sistemas computacionais especializados na proteção de outros sistemas. Como as redes são um meio essencial para o acesso aos componentes de um sistema de computação, a vigilância específica sobre o tráfego das redes pode resguardar a segurança de toda a instalação. Para este fim foram criados os NIDS, IDSs especializados na vigilância sobre o tráfego de redes.

Dentre as tarefas de um NIDS destacam-se as seguintes:

- Captura de pacotes – um NIDS tem que capturar pacotes da rede para realizar uma análise e determinar a presença de ameaça, ou ataque, no seu conjunto de dados.
- Comparação de padrões – é a tarefa principal do sistema e a que consome maior poder computacional. Cada byte do pacote deve ser confrontado com milhares de regras para determinar um provável ataque.

3.1 Snort – Descrição de operação

Snort [SNO06] é um NIDS capaz de realizar análise de tráfego em tempo real, seleção e registro de pacotes em redes TCP/IP. O Snort pode realizar identificações de protocolo, busca ou comparação de conteúdo, e pode ser utilizado para detectar uma variedade de ataques e *probes* (análises remotas ou sondagens). Por ser um sistema em software de fonte aberto [LAW02], o Snort recebe contribuições de programadores e especialistas em segurança de todo o mundo. Desde 1998, quando foi lançado, tem recebido constante atualização de seus algoritmos, de sua estrutura interna e de sua capacidade de identificar novos tipos e variantes de intrusões.

O Snort utiliza os recursos oferecidos pela biblioteca *libpcap* [TCP03], que fornece

monitoramento de pacotes (*packet sniffing*) em modo promíscuo. O modo de rede promíscuo é a configuração da interface de rede na qual todos os pacotes que nesta chegam são capturados, independente se este pacote é ou não destinado a este hospedeiro. A biblioteca *libpcap* foi escrita como parte de um programa maior, chamado *TCPDump* [TCP03], utilizado para decodificar, visualizar ou registrar pacotes em uma rede IP.

O Snort realiza detecção baseada em regras, sendo este um dos motivos que lhe proporcionam flexibilidade. As regras permitem configurar o sistema de detecção conforme requisitos de desempenho ou tipo de tráfego. O Snort permite a identificação de protocolos, de assinaturas e de anomalias em protocolos. Por meio de regras, pode-se especificar a detecção de uma condição complexa, e a partir daí determinar uma reação.

Snort inclui *plug-ins* em sua arquitetura como estratégia de flexibilidade. Um uso importante desta modularização é no tratamento que pode ser dado aos alertas resultantes da atividade do Snort como um NIDS. Como o número de alertas e registros pode tornar-se muito grande em redes sob ataque, alguma forma de classificação e tratamento automático adequado para cada situação torna-se necessária. Entretanto, o número elevado destas situações não permite um tratamento padronizado. Daí a importância da modularização, fazendo com que cada situação possa ser tratada de maneira particular. A modularização é importante igualmente para prover tratamento adequado aos diferentes regimes de tráfego encontrados na aplicação do Snort.

Snort é fornecido junto com uma coleção de regras, que inclui regras desativadas como forma de documentação. O site do Snort (http://www.snort.org) mantém um banco de dados com informações sobre todas as regras já publicadas. Ao ser instalado, o Snort gera um diretório onde são instaladas todas as regras conhecidas. Neste diretório as regras são agrupadas por afinidade, em arquivos cujo nome caracteriza sua aplicação. A Figura 11 mostra uma árvore de diretórios de uma instalação típica do Snort, com detalhe no diretório de regras (diretório *rules*) e parte de seu conteúdo.

Um arquivo de configuração, usualmente chamado de *snort.conf*, especifica quais dos arquivos de regras devem ser utilizados quando o Snort operar como um NIDS. Este arquivo também configura a modularização do Snort, ativando *plug-ins* seletivamente.

O Snort aplica-se a três principais usos:

- Monitoramento de pacotes;
- Registra de pacotes;
- NIDS.

A função de monitor de pacotes (*sniffer*), ativada através da opção "-v" na linha de comando, permite visualizar os pacotes coletados da rede na tela.

Como registrador de pacotes (*logger*) a coleta de pacotes é direcionada para um arquivo, para futura análise ou uso como padrões de tráfego, por meio da reprodução destes pacotes na rede. A opção de linha de comando "-l" seguida do nome de um arquivo destino ativam este modo e permitem ao Snort registrar neste arquivo o tráfego da rede.

Como um NIDS, o nome do arquivo de configuração (*snort.conf*), deve ser adicionado na linha de comando. Este arquivo contém a descrição de componentes opcionais que o Snort deverá ativar para esta função. Entre estes estão os conjuntos de regras de detecção e os pré-processadores

que são responsáveis pela geração dos alertas esperados de um NIDS. Em aplicações comerciais o Snort pode ser encontrado fazendo uso exclusivo do hardware de um servidor, associado a uma máquina *firewall* ou a um roteador. Quando não há restrições de investimento, aplica-se um NIDS para cada sub-rede, associa-se o Snort ao *firewall* e aos roteadores da rede

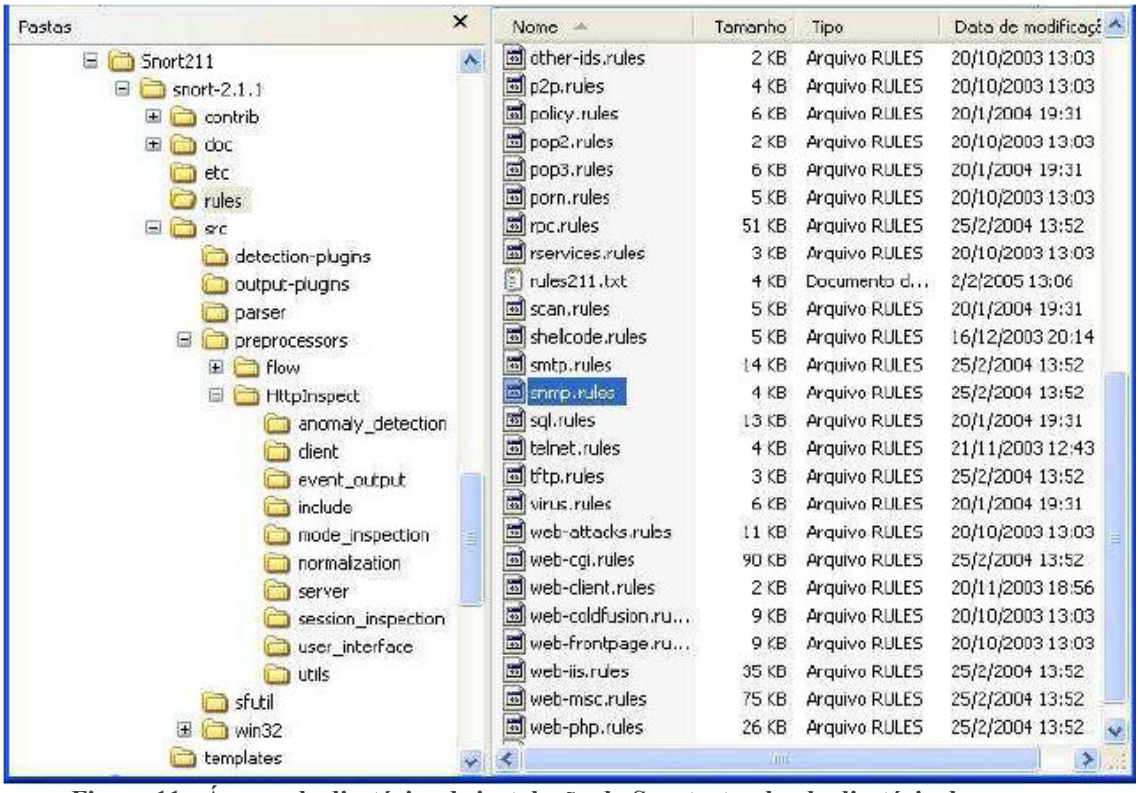


Figura 11 - Árvore de diretórios de instalação do Snort e trecho do diretório das regras.

Em instalações mais simples, como a apresentada na Figura 12, a rede interna é protegida por um *firewall* e um NIDS é colocado à frente deste, ou associado a este, antes da rede externa. Muitas vezes, nesta topologia, um roteador é ainda posicionado à frente do NIDS.

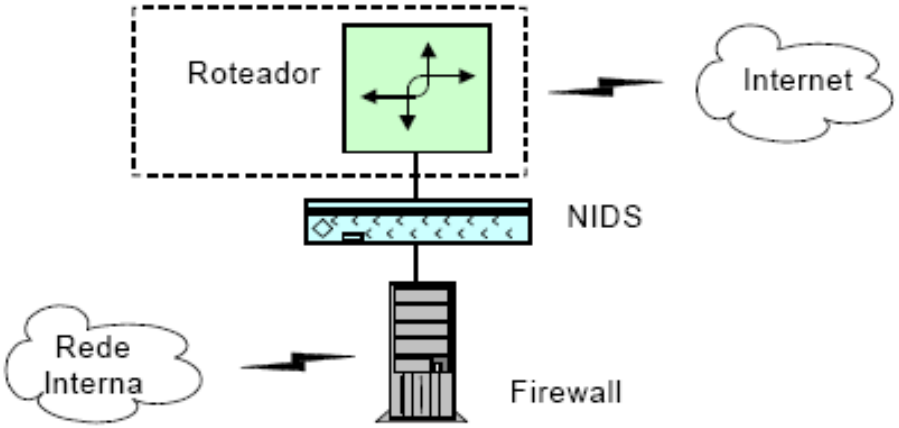


Figura 12 - Arquitetura simples com NIDS e roteador opcional.

Redes que ofereçam ao acesso externo uma das chamadas zonas desmilitarizadas (DMZ), acessam a rede externa por meio de um roteador com NIDS, conforme mostra a Figura 13. Uma zona desmilitarizada ou DMZ corresponde ao segmento ou segmentos de rede, parcialmente protegido, que se localiza entre redes protegidas e redes desprotegidas e contém todos os serviços e informações para clientes ou públicos. Um *firewall* conectado ao roteador, pelo lado interno,

distribui tráfego à DMZ e à rede interna por meio de respectivos NIDS adicionais. Neste caso se utiliza o Snort tanto na rede interna quanto na DMZ, pois podem existir ataques específicos a ambas as redes.

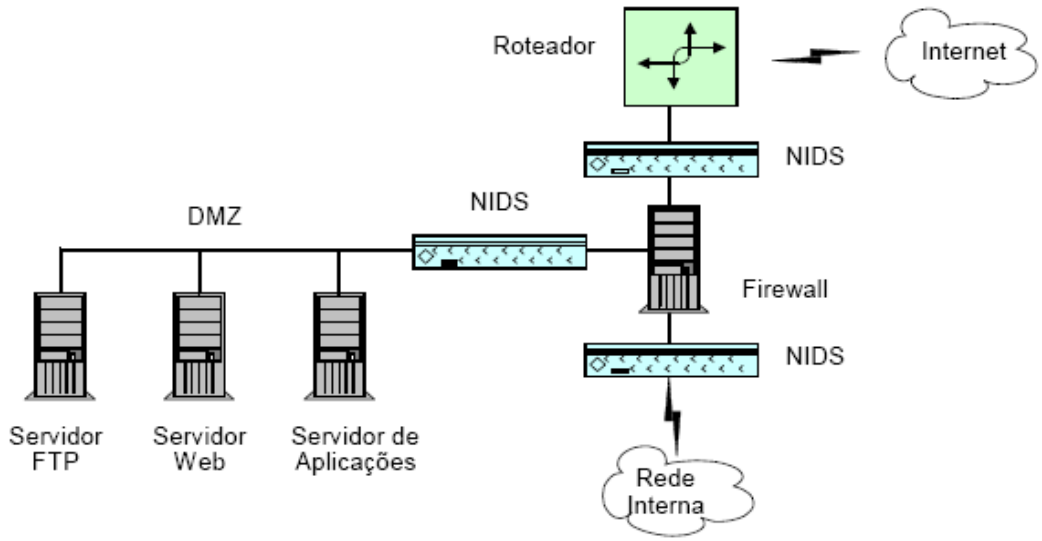


Figura 13 - Rede com DMZ Firewall e Snort.

3.2 Snort – O formato das regras

Para ilustrar o formato básico das regras aceitas pelo Snort, apresenta-se a seguir uma regra que detecta um ataque FTP. As regras do Snort são escritas em formato texto em uma única linha, e constituem-se de duas sessões: o cabeçalho e as opções.

- O cabeçalho da regra: alert tcp $EXTERNAL_NET any -> $HOME_NET 21
- As opções da regra: (msg: "FTP EXPLOIT wu-ftpd 2.6.0 site exec format string overflow Linux"; flow:to_server, established; content:"|31c031db31c9b046cd8031c031db|"; reference:bugtraq,1187; reference:cve,CAN-2000-0573; reference arachnids,287; classtype: attempted_admin; sid:344; rev4;)

A sintaxe usada no cabeçalho das regras compreende:

- **alert**: Formato utilizado para a saída. Os formatos de saída são: alert, log, pass, dynamic e activate.
- **TCP**: Protocolo em uso. Este campo pode aceitar os valores: TCP, UDP, IP e ICMP.
- **$EXTERNAL_NET**: Endereço IP de origem.
- **any**: Porta de origem.
- **->**: Direção do fluxo de dados. Neste caso, de $EXTERNAL_NET vindo de qualquer porta para $HOME_NET na porta 21.
- **$HOME_NET**: Endereço de destino.
- **21**: Porta destino.

O principal componente nas opções da regra é o seu conteúdo (do inglês, *content*). O campo *content "|31 C0 31 DB 31 C9 B0 46 CD 80 31 C0 31 DB|"* é comparado com o pacote em análise usando um algoritmo de comparação de cadeias de caracteres.

4 Implementação de NIDS em Hardware

Este Capítulo descreve inicialmente o SPP-NIDS (*Sea of Processors Platform for Network Intrusion Detection System*), uma implementação de NIDSs desenvolvida no trabalho de Mestrado de Luís Mieres Caruso [CAR05a], e utilizado como base para o desenvolvimento do presente Trabalho de Conclusão. A segunda parte do Capítulo apresenta um NIDS em hardware, comercial, com funções similares à proposta deste trabalho.

4.1 SPP-NIDS – Um NIDS em Hardware

A plataforma SPP-NIDS é uma implementação hardware-software de um NIDS, com suporte a regras do Snort. Esta plataforma não foi integrada com a rede externa, nem com sistemas de gerenciamento, o que é parte da contribuição deste trabalho, e será apresentado no Capítulo 5. O SPP-NIDS é viável para situações reais de detecção de intrusão de rede, pois:

- O sistema é escalável. Adicionar novas unidades de comparação ao projeto é uma questão de código HDL parametrizável;
- O número de regras que o sistema suporta é alterado pela reprogramação do banco de regras na memória de um processador;

4.1.1 Visão Geral da arquitetura do SPP-NIDS

O diagrama de blocos da plataforma e suas interfaces externas estão descritas na Figura 14 [CAR05b]. A arquitetura SPP-NIDS [CAR05b] é composta por um processador de controle IDS, um co-processador dedicado reconfigurável (cluster GIOIA na figura), um processador hospedeiro e interfaces para um enlace que conecta as redes externa e interna.

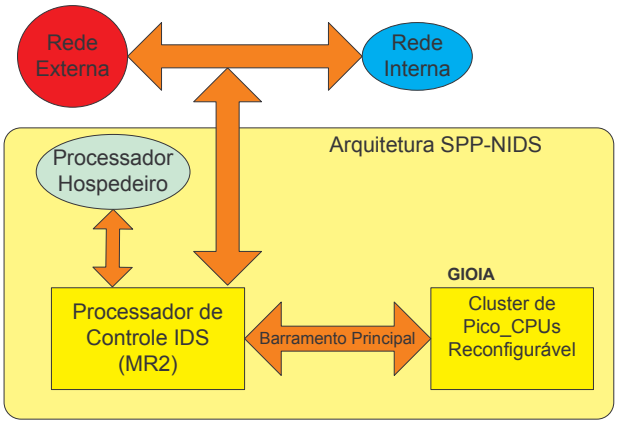


Figura 14 – Diagrama de blocos da arquitetura do SPP-NIDS.

O módulo principal do sistema é o co-processador dedicado reconfigurável chamado GIOIA. GIOIA é um *cluster* de processadores muito simples e idênticos, personalizados para comparação de regras. Cada um destes processadores simples é chamado de pico-CPU. Este co-processador foi projetado especificamente para realizar a comparação de padrões para aplicação das regras do Snort. Conectando pico-CPUs em paralelo obtém-se um determinado desempenho, o qual é escalável conforme o número de pico-CPUs.

O processador de controle IDS, na SPP-NIDS, é chamado de MR2. Ele é responsável pela inserção de regras na memória de cada pico-CPU, e pela monitoração e controle do GIOIA. Ele também é responsável pela interação com o processador hospedeiro, e com as redes interna e externa. O MR2 [CAL04] é um processador com organização Harvard que implementa um conjunto parcial de instruções do MIPS 2000, uma versão da arquitetura do MIPS I ™.

O processador hospedeiro é tipicamente um equipamento externo (computador), responsável pela inicialização do processador de controle IDS e do cluster GIOIA.

4.1.2 Estratégia de prototipação da SPP-NIDS

A arquitetura SPP-NIDS foi proposta em [CAR05a]. O primeiro protótipo de hardware deste sistema foi desenvolvido visando à participação no Xilinx Student Contest de 2005, cujas regras fixavam o uso da plataforma de prototipação Digilent Spartan-3 System Board para o projeto [XIL06]. Devido às restrições de entrada/saída de dados imposta por esta plataforma de prototipação, algumas alterações no projeto inicial da arquitetura do SPP-NIDS foram realizadas. Esta Seção descreve estas alterações e o resultado final do sistema prototipado.

Primeiramente, o kit Spartan-3 não contém uma interface de rede incorporada, como por exemplo, um módulo PHY Ethernet acoplado a um conector RJ-45. Mesmo se um módulo PHY estivesse disponível, o módulo MAC Ethernet e a implementação da pilha IP ocupariam uma grande parte do FPGA Spartan-3 de duzentas mil portas lógicas equivalentes da placa, inviabilizando a implementação do sistema SPP-NIDS no mesmo dispositivo.

Desta forma, o comportamento da rede é emulado, usando uma parte da memória de dados do MR2, contendo pacotes IP que são lidos por rotinas especializadas de software executadas no processador MR2. Ambas as memórias de dados e instruções são implementadas com Block RAMs internas ao FPGA.

Outra característica do protótipo implementado é que o processador hospedeiro é de fato um computador PC conectado à plataforma de prototipação via interface serial, usando um protocolo compatível com o padrão RS-232. Esta interface permite ao hospedeiro reinicializar e parar o processador MR2, além de permitir a carga da memória de instruções do MR2 com o software de gerência do cluster GIOIA e a carga da memória de dados do MR2 com as regras do Snort que serão enviadas a cada pico-CPU e pacotes IP que emulam o comportamento da rede. O hospedeiro ainda utiliza a interface serial para receber informações sobre tentativas de invasão nos pacotes IP.

No protótipo, o computador hospedeiro não foi utilizado para o envio de pacotes reais ao sistema, dada à baixa velocidade da interface serial. Como o objetivo do protótipo era validar o módulo de comparação do cluster GIOIA, a utilização de uma interface muito mais lenta que uma interface de rede real invalidaria o sistema.

4.1.3 A estrutura do protótipo SPP-NIDS

A Figura 15 apresenta a estrutura do protótipo da arquitetura SPP-NIDS. O cluster GIOIA é mapeado na memória de dados do processador, utilizando os endereços mais altos da memória. O acesso ao cluster é realizado via operações de escrita e leitura na memória do MR2.

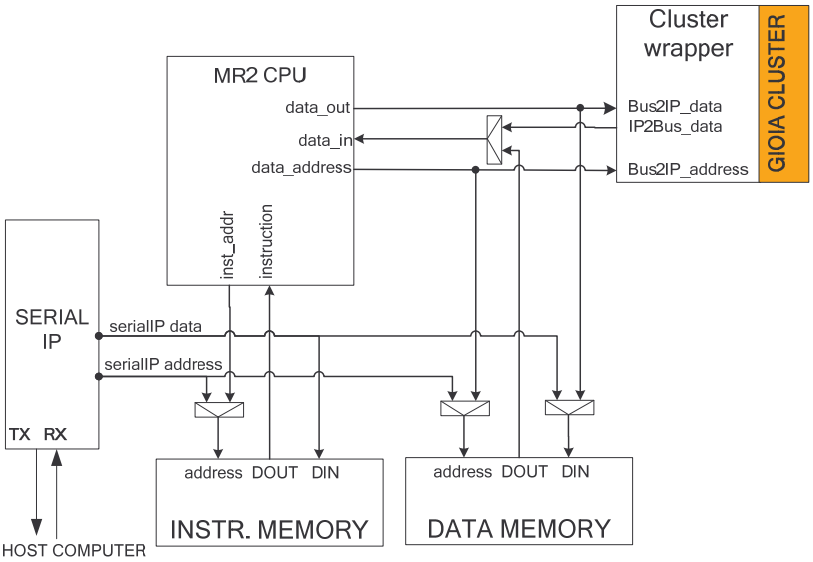


Figura 15 – Estrutura geral do protótipo da plataforma SPP-NIDS.

As Seções a seguir descrevem a pico-CPU e cada um dos módulos no protótipo do sistema SPP-NIDS.

4.1.4 A pico-CPU

A função da pico-CPU é verificar se certos padrões (chamados de regras, no Snort) ocorrem em alguma seção do tráfego de entrada da rede. As pico-CPUs armazenam em memórias RAM internas os padrões com os quais o tráfego da rede deve ser comparado. O uso de memória RAM para armazenar os padrões de ataque concede à pico-CPU sua propriedade de reconfigurabilidade. Esta memória foi implementada com um bloco de memória RAM dupla porta (do inglês, *Dual Port RAMs* ou *DPRAMs*) presente no FPGA (*Block Select RAMs* ou *BRAMs*). Já que o Snort possui um conjunto de regras em constante evolução, a reconfiguração é uma das principais características na plataforma SPP-NIDS. Sempre que o Snort disponibilizar um novo conjunto de regras, estas poderão ser imediatamente adicionadas ao SPP-NIDS, bastando atualizar a memória de dados do MR2.

Conforme ilustrado na Figura 16, as memórias *DPRAMs* armazenam subconjuntos do total dos padrões de comparação, organizados em classes (TCP, UDP, IP e ICMP). Os padrões serão armazenados seqüencialmente e são acessados byte a byte. As *DPRAMs* são organizadas em 3 regiões:

- O bloco de registros contém as informações de acesso e controle de cada padrão (Regra). Cada registro contém quatro campos: endereço inicial do padrão na memória (E1 cx py), número de bytes do padrão (T cx py), identificação do padrão (ID cx py) e estado da comparação (S cx py). O estado da comparação (S cx py) é um contador, que é incrementado a cada comparação positiva entre um byte da carga útil (*payload*) e o padrão em comparação.
- O bloco de vetores armazena os endereços correspondentes ao primeiro registro de cada classe (Protocolo). Desta forma, cada classe pode ter um número arbitrário de registros.
- O bloco de padrões armazena os padrões. Este bloco contém diversos sub-blocos, um para cada padrão armazenado.

A divisão de classes adotadas coincide com a divisão de protocolos reconhecidos pelo Snort quais sejam: TCP, UDP, ICMP e IP.

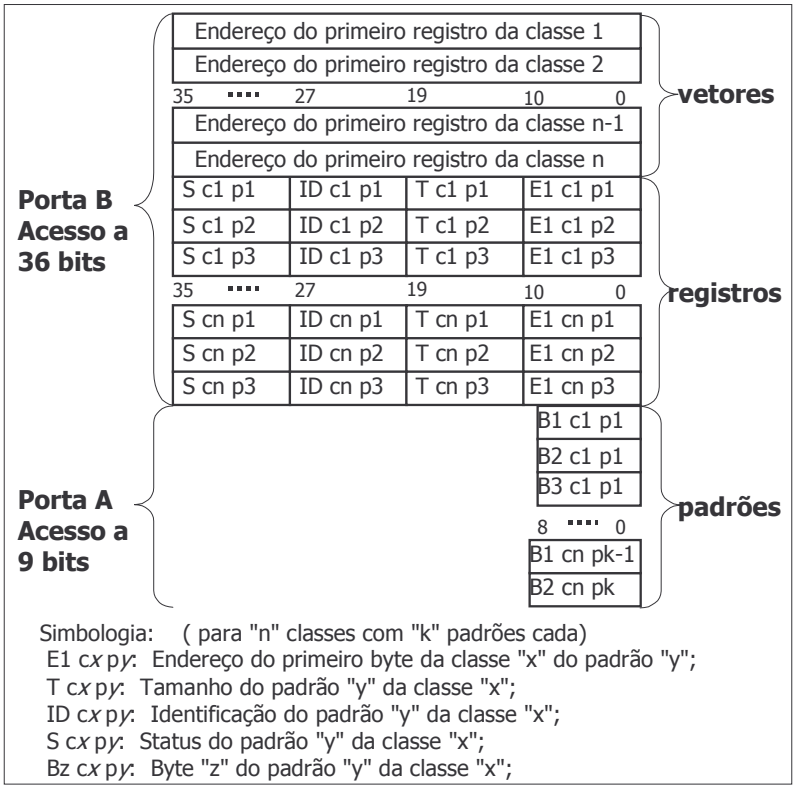
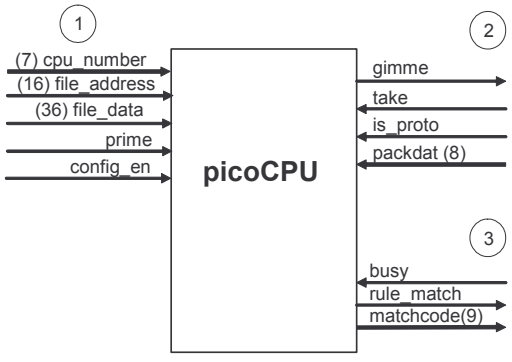


Figura 16 - Organização das memórias DPRAMs utilizadas pelas pico-CPUs.

A Figura 17 mostra o conjunto de sinais de entrada/saída de dados e controle de uma pico-CPU. A interface de configuração (índice 1 na Figura 17) permite a carga das regras na memória interna da pico-CPU. Esta configuração é realizada pelo processador MR2. Todas as pico-CPUs dividem um mesmo barramento de configuração e cabe ao sinal ***cpu_number*** (número único para cada pico-CPU) ditar qual pico-CPU será configurada por vez. Os sinais ***file_address*** e ***file_data*** correspondem aos barramentos de endereços e dados, respectivamente. Os sinais ***prime*** e ***config_en***, quando ativos, indicam que o barramento de configuração está sendo utilizado.



1 – Interface de Configuração;
2 – Interface de Pacotes; 3 – Interface de Intrusão

Os valores entre parênteses indicam o número de bits de cada barramento.

Figura 17 - Conjunto de sinais de entrada/saída de dados e controle de uma pico-CPU.

A interface de pacotes (índice 2) recebe os dados de entrada da rede. O sinal ***gimme*** é usado pela interface de pacotes para descobrir se a pico-CPU está ociosa, e desta forma pronta para processar um novo byte de dados. Os sinais ***take***, ***is_proto*** e ***pakdat***, são usados para o controle de transferência de novos dados à pico-CPU. Quando o sinal ***is_proto*** é ativado, indica o início de um novo pacote que será enviado à pico-CPU.

A interface de intrusão (índice 3) permite que a pico-CPU informe ao processador MR2 a detecção de um provável ataque à rede. A ocorrência de um ataque é indicada pela ativação do sinal ***rule_match*** e o número da regra do Snort que foi acionada naquela pico-CPU é informado no barramento ***matchcode***. Se o sinal ***busy*** for ativado, a pico-CPU deve manter o ***matchcode*** estável até que o processador de controle IDS (MR2) libere a sua operação.

Em linhas gerais, uma pico-CPU implementa um algoritmo que pode ser descrito como dois laços aninhados. O laço externo é responsável pela amostragem de todos os caracteres de um pacote, um a um. O laço interno compara cada caractere amostrado com um dos bytes de todas as regras disponíveis nesta pico-CPU. Uma dentre quatro possibilidades pode ocorrer, durante a comparação de um caractere amostrado e um byte de alguma das regras:

1. Sem acerto, com o *ponteiro de regra*[1] no início da regra (significa que o campo *S cx py* na Figura 17 permanece com valor em zero), neste caso nenhuma modificação de estado ocorre nas DPRAMs;
2. Acerto, sem que esteja no final da regra. Neste caso o ponteiro de regra deve ser adiantado em uma posição na regra (incremento no campo *S cx py*);
3. Sem acerto, mas comparações anteriores fizeram o ponteiro de regra avançar na regra. Neste caso o ponteiro deve ser reinicializado à posição de início;
4. Acerto, com o ponteiro de regra no último byte da regra (campo *S cx py* igual ao campo *T cx py*), neste caso o sinal rule_match é ativado informando a ocorrência de um ataque.

A situação padrão é a do item 1 acima, executada em um ciclo de relógio. As situações 2 e 3 levam dois ciclos de relógio para serem executadas. A situação 4 corresponde a uma detecção de regra. O tratamento desta situação depende da CPU de controle IDS e do tamanho do cluster GOIA. Já que as detecções são armazenadas, ocorre pouca perda de desempenho no caso de detecção de ataques.

Como um estudo de caso de requisitos de desempenho, considere-se um tempo médio de 1,5 ciclos relógio para processar cada caractere em um pacote (p_t), e uma dada pico-CPU trabalhando a 200 MHz (f) com um conjunto de 20 regras (n_r). A vazão de dados (T_p) excede 53 Megabits por segundo (Mbps), o que pode ser calculado com a equação $\mathbf{T_p = (f * 8)/(p_t * n_r)}$. A constante 8 representa o número de bits em um caractere.

A Figura 18 ilustra o bloco de dados da pico-CPU, uma implementação centrada em uma RAM de dupla porta (DPRAM ou Dual Port RAM). Durante o processo de configuração, os dados são carregados na DPRAM usando a interface de configuração descrita anteriormente.

O primeiro byte de um pacote (sinalizado pelo sinal ***is_proto***, vide Figura 18) é a informação de cabeçalho, definindo o protocolo utilizado (IP, ICMP, TCP, UDP). Este byte inicializa o endereço da porta B (ADDRB), que é um ponteiro para um conjunto de *descritores de regras* referente ao protocolo do pacote em análise. Cada descritor de regras, presente na saída DOB da DPRAM contém quatro campos, três fixos e um variável. Os três campos fixos são: ***E1***, o endereço do início da regra; ***size***, o número de bytes presentes na regra; ***ID***, o identificador único de cada regra. O campo variável é chamado de ***Displ***, que representa o deslocamento nos bytes da regra a partir do seu início. A saída da porta A na DPRAM é a leitura da posição de memória do endereço obtido pela soma de E1 e ***Displ*** como descrito pela Figura 18. Quando a saída da porta A é igual ao

[1] *Ponteiro de regra* indica na memória de padrões a regra em comparação.

byte do pacote corrente, um acerto (***match***) é obtido.

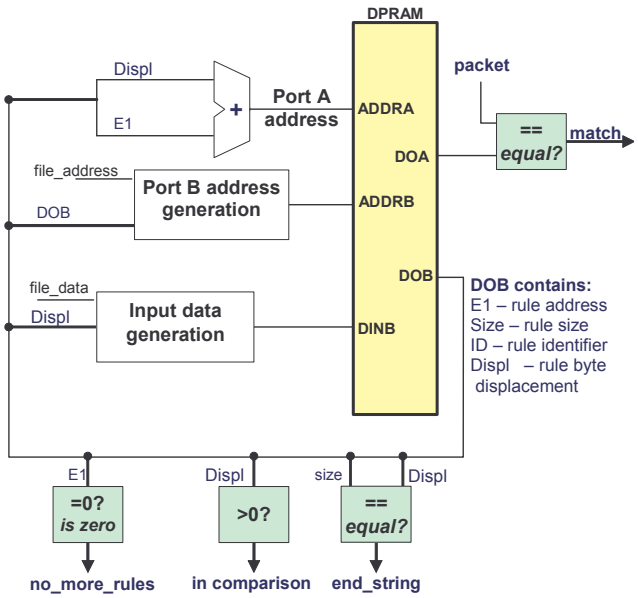


Figura 18 – Estrutura interna do bloco de dados de uma pico-CPU.

Se um acerto não ocorrer (***match*** = 0) e a regra estava no modo de comparação (sinal ***in_comparison*** > 0), o campo ***Displ*** é reinicializado à zero. Se um acerto ocorrer e o campo ***Displ*** é igual ao tamanho da regra (**size**) o sinal ***end_string*** é ativado, indicando a detecção de um ataque. Se um acerto ocorrer, mas o final da regra ainda não foi atingido o sinal ***Displ*** é incrementado. Quando todas as regras foram verificadas, o sinal ***no_more_rules*** é ativado, liberando a pico-CPU para receber novos bytes a serem comparados.

4.1.5 O cluster GIOIA

O Snort realiza a separação das regras a serem comparadas por protocolo. Desta forma, o software irá realizar comparações com um conjunto de regras menor do que o total de regras. O Snort na sua versão 2.11[SNO06] separa suas regras em quatro protocolos principais: IP, TCP, UDP e ICMP. O número de regras por protocolo presentes no Snort versão 2.11 é descrito na Tabela 1. O conjunto de regras presentes no Snort é atualizado freqüentemente, e o maior conjunto de regras pertence ao protocolo TCP. Estas regras podem ser subdivididas em subclasses, como por exemplo, filtros de endereços IP. Como o cluster GIOIA foi desenvolvido para dar suporte ao conjunto de regras do Snort, este também possui regras nos quatro protocolos citados acima. Um NIDS operando sobre uma rede 100 Mbps, deveria comparar cada dado recebido contra 1784 regras no intervalo entre dois bytes: 80 ns. Este desempenho é inatingível quando executando em software, justificando a implementação em hardware dedicado.

Tabela 1 – Número de regras por protocolo no Snort v.2.11.

Protocolo	Número de Regras
TCP	1784
UDP	165
IP	44
ICMP	131
TOTAL	2124

Um NIDS de qualidade deve ser capaz de comparar cada caractere de entrada da rede contra, no mínimo, quarenta e quatro regras (que equivalem ao menor grupo de regras de um determinado protocolo) no tempo entre duas recepções de caracteres na rede. Este limite estabelece um requisito

mínimo de desempenho no projeto para o *cluster* GIOIA.

Da mesma forma que as pico-CPUs, o cluster possui três grupos de sinais, implementando as três interfaces distintas de cada pico-CPU:

- Interface de configuração – responsável pela carga das regras do Snort no cluster;
- Interface de pacotes – responsável pela recepção dos dados extraídos da rede;
- Interface de detecção de intrusão ou simplesmente interface de intrusão – responsável pela sinalização de ataques ao processador de controle IDS.

A descrição das interfaces acima é a mesma da Seção 4.1.4. A Figura 19 mostra a interface externa e a organização interna do cluster GIOIA, abstraindo a interface de configuração.

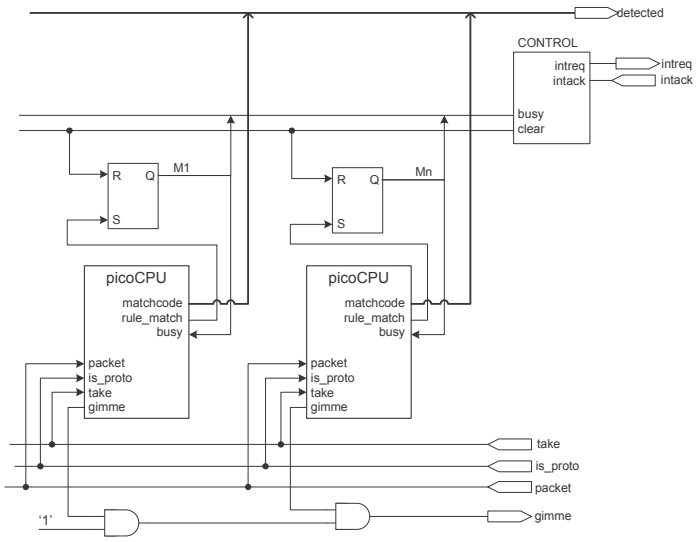


Figura 19 – Interface interna do cluster e a interface com a CPU de controle IDS.

A interface de pacotes pode receber um novo caractere somente se todas as pico-CPUs estiverem livres para processar este novo caractere. O sinal ***gimme*** é usado para informar esta condição das pico-CPUs. Esta é a razão pela qual o cluster precisa estar balanceado na distribuição de regras de suas pico-CPUs, de tal forma que os tempos de processamento de cada pico-CPU sejam aproximadamente idênticos, reduzindo a ociosidade das pico-CPUs.

Quando uma pico-CPU detecta um ataque, o sinal ***rule_match*** é ligado, ativando o flip-flop de detecção correspondente (um dos flip-flops de **M1** a **Mn**). Quando um dado flip-flop de detecção é ativado, um sinal de ***busy*** é gerado para a pico-CPU correspondente, travando esta. A unidade de controle do cluster realiza uma varredura *daisy-chain* em todos os registradores de detecção, procurando qual pico-CPU detectou um ataque. Quando um ataque é detectado, um sinal de interrupção é gerado para o processador MR2 e o código do ataque encontrado (chamado de ***matchcode***) é armazenado em um registrador do cluster (não mostrado na Figura 19). Ao final desta operação todo o cluster é liberado através do sinal *clear*, desta forma a pico-CPU que detectou o ataque e estava travada é liberada e volta a sua operação normal. O tempo máximo que uma pico-CPU fica travada e não realiza seu algoritmo de detecção é, em ciclos de relógio, igual ao número de pico-CPUs no cluster menos um. O processador MR2, ao realizar o tratamento da interrupção gerada pelo cluster, salva o código do ataque que estava armazenado em registrador e libera a varredura *daisy-chain* com um sinal de ***int_ack***.

4.1.6 O wrapper do cluster GIOIA

O *wrapper* do cluster tem a função de adaptar os sinais para o barramento de dados do processador de controle IDS, e está ilustrado na Figura 20. Esta adaptação ao barramento permite o acesso às três interfaces do cluster usando, operações de entrada/saída mapeadas em memória.

O processo de configuração do cluster começa com o MR2 escrevendo dados no banco de registradores interno do *wrapper*. Após detectar uma operação de escrita nos registradores de configuração, o *wrapper* gera um pulso no sinal ***config_en***. Este mesmo procedimento é realizado para prover dados da rede para o cluster. Uma operação de escrita nos registradores destinados à transmissão de pacotes ao cluster gera pulsos nos sinais ***take*** e ***is_proto***. É importante salientar que não há necessidade de desativar os sinais de controle via software, sendo esta ação executada no hardware do *wrapper*.

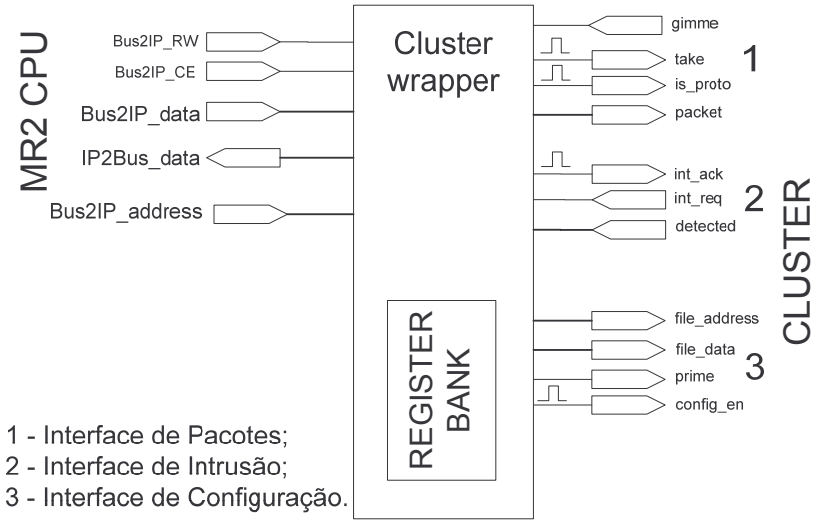


Figura 20 – Interfaces e estrutura parcial do wrapper do cluster.

A implementação do SPP-NIDS descrita a partir da Seção 4.1.2, coleta os dados de ataques através de *polling* e não interrupções como descrito acima. O software do MR2 monitora a detecção de ataques e, quando um ataque é identificado, executa ações de tratamento deste. Após tais ações de tratamento serem iniciadas, o MR2 gera um pulso no sinal ***int_ack*** a fim de liberar a unidade de controle do cluster.

4.1.7 O módulo serial

O módulo serial é responsável por prover comunicação entre o usuário trabalhando em um computador hospedeiro e o sistema de detecção de intrusão. Esta comunicação é realizada via um protocolo serial padrão RS-232.

O computador hospedeiro é responsável por enviar: o software a ser executado pelo processador MR2, as regras às pico-CPUs, e os dados capturados na rede – caracteres a serem comparados. O computador hospedeiro recebe do sistema o número de ataques detectados e o número da regra de cada ataque para futura análise.

4.1.8 O processador MR2

O processador que controla o cluster é chamado de MR2 [CAL04]. Ele é um processador Harvard de 32 bits que implementa parcialmente uma arquitetura do conjunto de instruções do processador MIPS 2000. O simulador SPIM [LAR05] é usado para validar o software que interage

com o cluster de pico-CPUs. O subconjunto de instruções do MIPS utilizado no MR2 aparece na Tabela 2.

Tabela 2 – Instruções do MR2, um subconjunto de instruções do MIPS 2000 [CAL04].

MR2 INSTRUCTIONS	
ADDU Rd, Rs, Rt	SB Rt, Imed16(Rs)
SUBU Rd, Rs, Rt	SW Rt, Imed16(Rs)
AND Rd, Rs, Rt	SLT Rd, Rs, Rt
OR Rd, Rs, Rt	SLTU Rd, Rs, Rt
XOR Rd, Rs, Rt	SLTI Rt, Rs, Imed16
SLL Rd, Rt, shamt	SLTIU Rt, Rs, Imed16
SRL Rd, Rt, shamt	BEQ Rs, Rt, label
ADDIU Rt, Rs, Imed16	BGEZ Rs, label
ANDI Rt, Rs, Imed16	BLEZ Rs, label
ORI Rt, Rs, Imed16	BNE Rs, Rt, label
XORI Rt, Rs, Imed16	J label
LUI Rt, Imed16	JALR Rs, Rd
LBU Rt, Imed16(Rs)	JAL label
LW Rt, Imed16(Rs)	JR Rs

4.2 FORCE10 P-Series

O único produto comercial encontrado pelos Autores deste trabalho que se assemelha à arquitetura proposta neste Trabalho de Conclusão é a plataforma P-Series da empresa *Force10* [FOR06]. Esta arquitetura é baseada em uma tecnologia chamada de DPI (*Dynamic Parallel Inspection*), ou Inspeção Dinâmica e Paralela, que permite que milhares de regras sejam verificadas sobre cada byte de um pacote. O produto funciona como um firewall, baseado nas regras do Snort. Sua estrutura funciona como filtros no tráfego da rede, como mostrado na Figura 21.

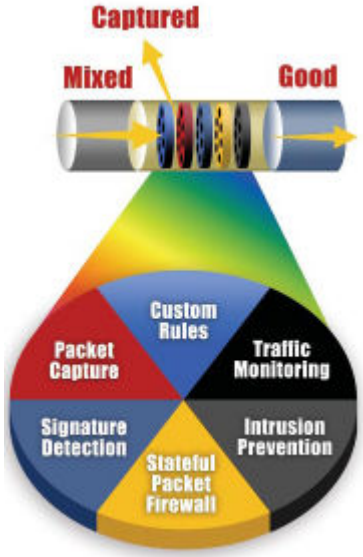


Figura 21 - Modelo de filtros usado na plataforma P-Series.

Segundo a empresa, o produto possui suporte às seguintes aplicações:

- Detecção e prevenção de intrusão de redes à taxas de 1 e 10 Gbps;
- Assistência na inspeção e flexibilidade na captura de dados para aplicações de monitoramento eletrônico;
- Interface de aplicação aberta para segurança de rede personalizada e aplicações de monitoramento de rede.

As características da plataforma P-Series, conforme apresentadas pela empresa, são as seguintes:

- Inspeção de pacotes com suporte a *Jumbo frames*, em uma rede de até 10 Gbps;
- Desempenho previsível desconsiderando as condições de tráfego na rede ou número de regras;
- Suporte à dois modos de operação, um ativo na linha e outro passivo de monitoramento/captura;
- Escrita de regras dinamicamente direto no hardware, o que permite uma maior flexibilidade na resposta ao tráfego malicioso;
- Compatível com os softwares Snort, Bro, Ethereal e Tcpdump.

A Figura 22 mostra os produtos P1 e P10, ambos usando arquitetura da plataforma P-Series, da empresa Force10.

Figura 22 - Firewall P-Series, da empresa Force10.

5 PrimeSec - Um NIDS em Hardware Multiprocessado

Nesta Seção é apresentada a arquitetura, denominada *PrimeSec*, desenvolvida durante este Trabalho de Conclusão. O objetivo desta arquitetura é garantir uma comunicação segura entre uma rede interna e outra externa, utilizando como estrutura de filtro o co-processador de comparação GIOIA.

5.1 Visão geral da arquitetura do PrimeSec

A arquitetura do PrimeSec, assim como o SPP-NIDS, é baseada na utilização do *cluster* GIOIA como estrutura principal (um co-processador de comparação). Diferentemente do SPP-NIDS, o projeto do PrimeSec pressupõe maior disponibilidade de recursos, incluindo área em silício, módulos PHY Ethernet, etc. Estendendo-se a arquitetura do SPP-NIDS obtém-se a base do sistema PrimeSec.

O desenvolvimento do sistema PrimeSec foi dividido em projetos de hardware e software. O projeto de hardware compreende o reuso de propriedade intelectual (MAC Ethernet, processador Plasma e o cluster GIOIA) e o desenvolvimento de módulos para a integração do sistema. A Figura 23 mostra a arquitetura interna do PrimeSec. O sistema compreende três processadores Plasma [PLM06], o cluster de pico-CPUs GIOIA [CAR05a], e dois MAC Ethernet 10/100 Mbps [MOH06].

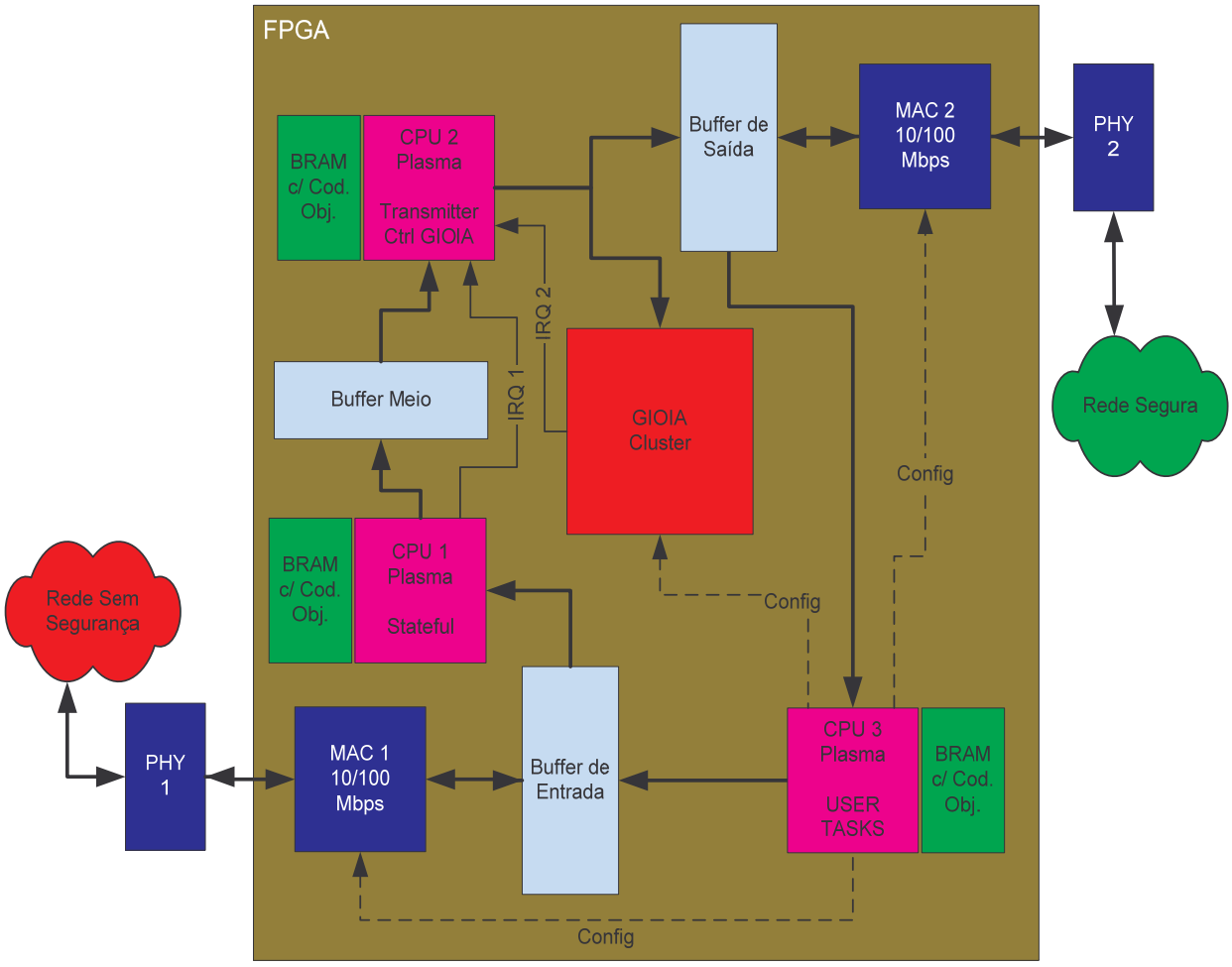


Figura 23 – Visão geral da arquitetura da plataforma PrimeSec.

As próximas Seções irão abordar, com mais detalhes, cada um dos módulos descritos acima.

5.2 O módulo MAC Ethernet 10/100 Mbps

O módulo MAC utilizado neste Trabalho de Conclusão é uma adaptação do módulo MAC Ethernet 10/100 Mbps do OpenCores [MOH06]. Este módulo está ligado de forma nativa ao barramento Wishbone [HER02] do Opencores. Este módulo MAC Ethernet foi adaptado, para se adequar ao contexto do projeto TETHA, no qual ambos os Autores estão inseridos. Nesta adaptação os sub-módulos referentes ao barramento Wishbone foram excluídos, utilizando-se apenas os principais sub-módulos. A adaptação limitou-se ao uso do modo de rede do tipo full-duplex, não tendo sido adaptado até o presente o momento o modo half-duplex.

Nesta Seção iremos abordar a estrutura deste MAC Ethernet adaptado, definindo seus sub-módulos.

5.2.1 Visão geral da arquitetura do módulo MAC Ethernet

O MAC é dividido em sete sub-módulos principais: módulo de transmissão (TxMAC), módulo de recepção (RxMAC), módulo de controle (MAC Control Module), módulo de status (MAC Status Module), interface MII (MII Interface), banco de registradores (MAC Register Bank) e controlador de dados (Data Control), sendo este último desenvolvido pela equipe de pesquisa do projeto TETHA. O MAC Ethernet possui duas interfaces de comunicação, uma com a camada física padrão Ethernet (PHY) e outra com um módulo que irá utilizar o MAC como módulo de recepção/transmissão de dados. A Figura 24 mostra a visão geral da arquitetura do MAC Ethernet.

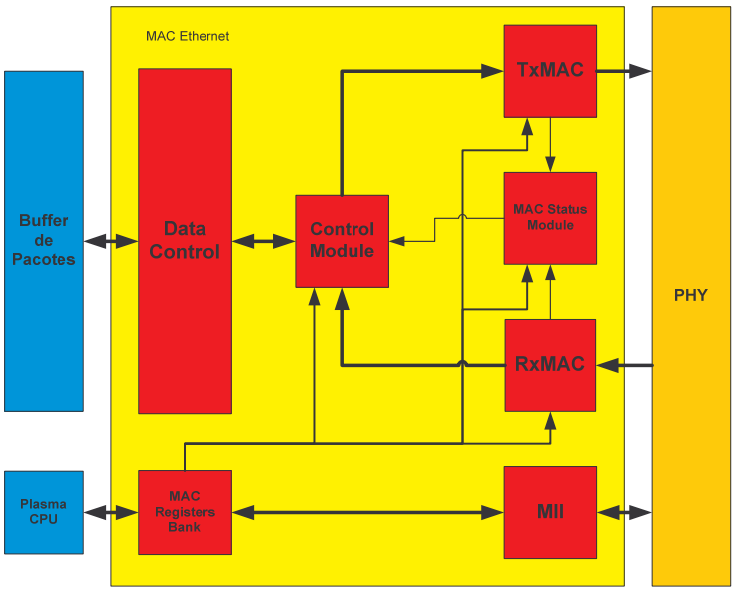


Figura 24 - Visão geral da arquitetura do MAC Ethernet e seu contexto de uso

5.2.2 Descrição das interfaces externas do módulo MAC Ethernet

A interface com a camada física do padrão Ethernet (PHY), como mostra a Figura 25, é efetuada com os seguintes sinais:

- **MtxClk** – clock gerado pelo PHY para sincronizar a transmissão;
- **Collision** – informa ao MAC a detecção de uma colisão no padrão CSMA-CD;
- **CarierSense** – informa ao MAC a detecção da portadora no padrão CSMA-CD;
- **MTxEn** – informa ao PHY a existência de um *nibble* válido para transmissão em MTxD;
- **MTxErr** – informa ao PHY a existência de um erro na transmissão dos dados;

- **MTxD** – transmite um *nibble* de dados ao PHY;
- **MDIO** – dados transferidos entre o módulo MII e o PHY;
- **MDC** – clock de sincronização entre o módulo MII e o PHY;
- **MRxClk** – clock gerado pelo PHY para sincronizar a recepção;
- **MRxErr** – informa ao MAC a existência de um erro na recepção dos dados;
- **MRxDv** – informar ao MAC a existência de um *nibble* válido para recepção em MRxD;
- **MRxD** – recebe um *nibble* de dados do PHY.

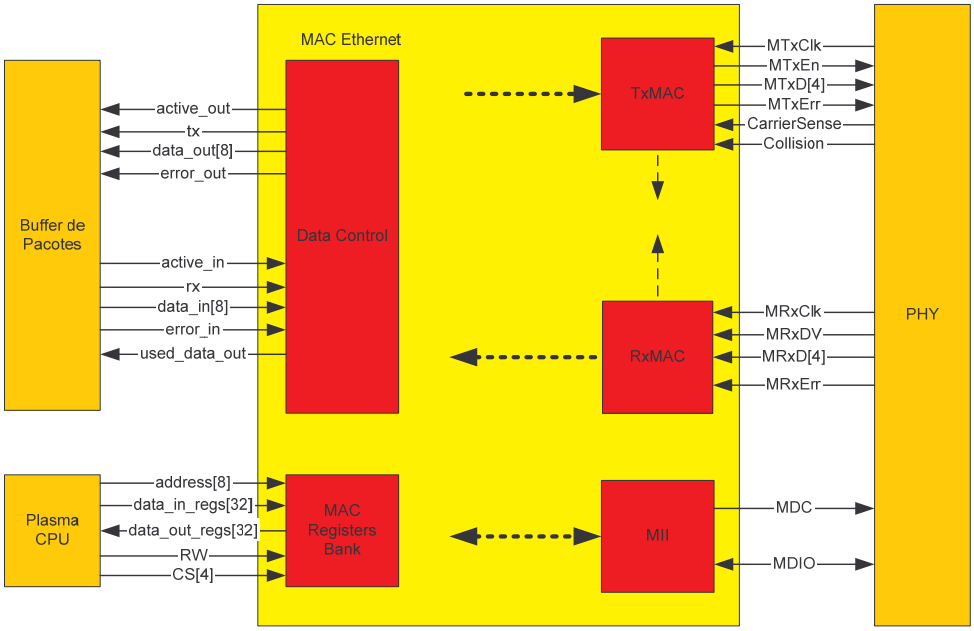


Figura 25 – Interface com a camada física Ethernet PHY e a interface com os módulos internos do sistema.

A interface com os demais módulos internos no sistema é baseada em um protocolo de handshake em 8 bits de dados. A configuração dos sub-módulos do MAC é realizada via banco de registradores. Os sinais utilizados neste protocolo são os seguintes:

- **active_in** – Utilizado para informar o MAC uma janela de transmissão de um pacote;
- **rx** – Utilizado para informar o MAC a existência de um novo byte válido no barramento de dados;
- **used_data_out** – Utilizado para informar o módulo que está enviando dados, que o byte presente no barramento de dados já foi consumido pelo MAC;
- **data_in[8]** – Utilizado para enviar um byte ao MAC;
- **error_out** – Utilizado para informar o MAC um erro na transmissão dos dados;
- **active_out** – Utilizado para informar ao módulo interno uma janela de recepção de um pacote;
- **tx** – Utilizado para informar ao módulo interno a existência de um novo byte no barramento de dados ;
- **data_out[8]** – Utilizado para enviar um novo byte ao módulo interno;
- **error_in** – Utilizado pelo MAC para informar um erro ao módulo interno;
- **data_out_regs[32]** – Utilizado para efetuar uma leitura no banco de registradores;

- **data_in_regs[32]** – Utilizado para efetuar uma escrita no banco de registradores;
- **RW** – Utilizado para informar uma leitura ou uma escrita no banco de registradores;
- **CS[4]** – Utilizado para selecionar um byte na palavra de 32 bits (quatro bytes);
- **ADDR[4]** – Utilizado para endereçar o banco de registradores.

5.3 O Processador Plasma

O processador Plasma é uma implementação da arquitetura MIPS R3000. Os únicos modos de instrução MIPS I ™ não suportados pelo Plasma são as exceções e as operações de leituras e escritas desalinhadas. O Plasma possui suporte a interrupções.

A Figura 26 representa uma visão geral do Plasma. Sua arquitetura possui três blocos principais: A CPU propriamente dita, denominada **mlite_cpu**, um bloco de memória interna, e uma interface serial chamada de UART. O Plasma possui um barramento de comunicação externo para uma expansão de memória, utilizando uma memória externa. Este barramento é o mesmo utilizado na comunicação entre a CPU **mlite_cpu** e o bloco de memória interna. O processador ainda possui uma interface serial e um barramento de propósito geral chamado de GPIO para a comunicação com outros dispositivos. Após a figura, segue uma breve descrição das interfaces externas ao processador Plasma.

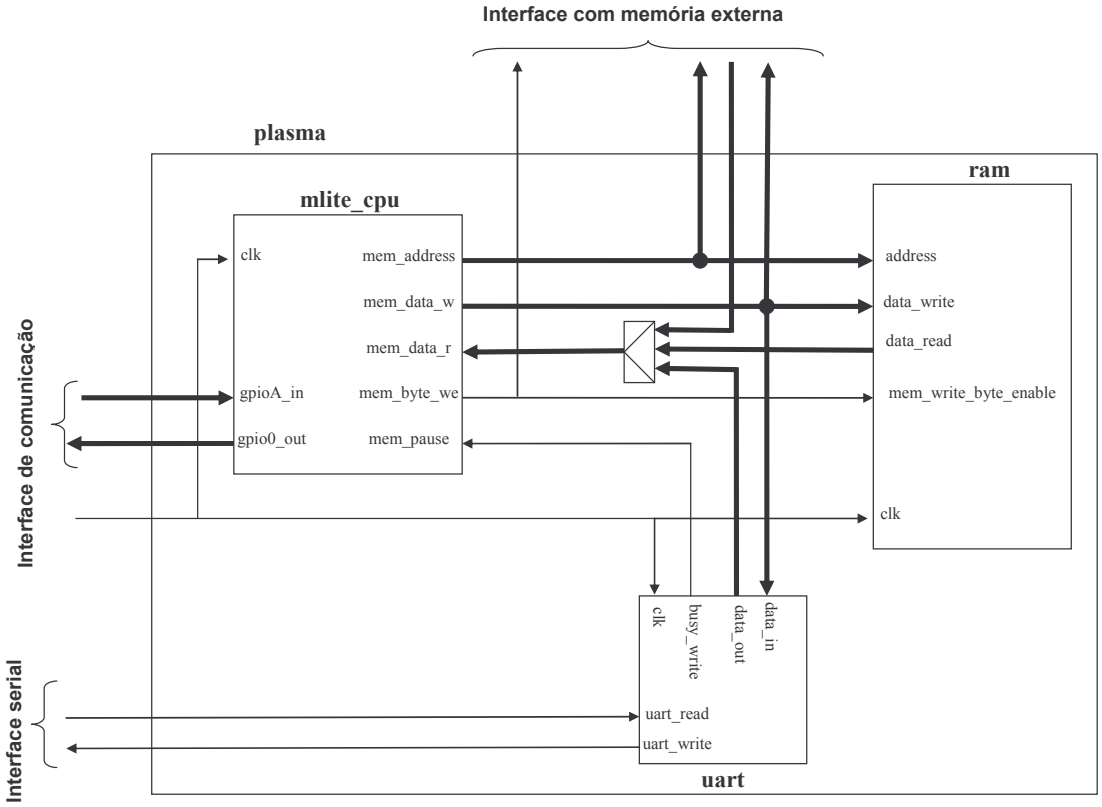


Figura 26 - Visão da arquitetura do processador Plasma e suas interfaces externas.

A interface de comunicação permite a um barramento externo realizar operações de entrada/saída em dispositivos externos ao processador, além de receber os sinais de interrupção (os dois bits mais altos no barramento gpioA_in).

A interface serial possibilita uma comunicação serial com qualquer dispositivo externo, como por exemplo um PC, para realizar operações de depuração.

5.4 O cluster GIOIA

O cluster GIOIA atua no sistema como um co-processador de detecção de padrões aos moldes do sistema SPP-NIDS. Um processador Plasma processa o pacote de entrada da rede e o repassa byte à byte ao GIOIA para comparação. Maiores detalhes sobre a estrutura do cluster GIOIA e seus métodos de comparação são descritos na Seção 4.1.5.

5.5 Visão geral das interfaces internas do PrimeSec

O sistema PrimeSec foi dividido em seis interfaces principais, as interfaces de:

- recepção de pacotes da rede sem segurança;
- verificação de pacotes;
- transmissão de pacotes à rede segura;
- recepção de pacotes da rede segura;
- configuração e monitoração do sistema;
- transmissão de pacotes à rede sem segurança.

Estas interfaces estão descritas na Figura 27 e serão detalhadas nesta Seção.

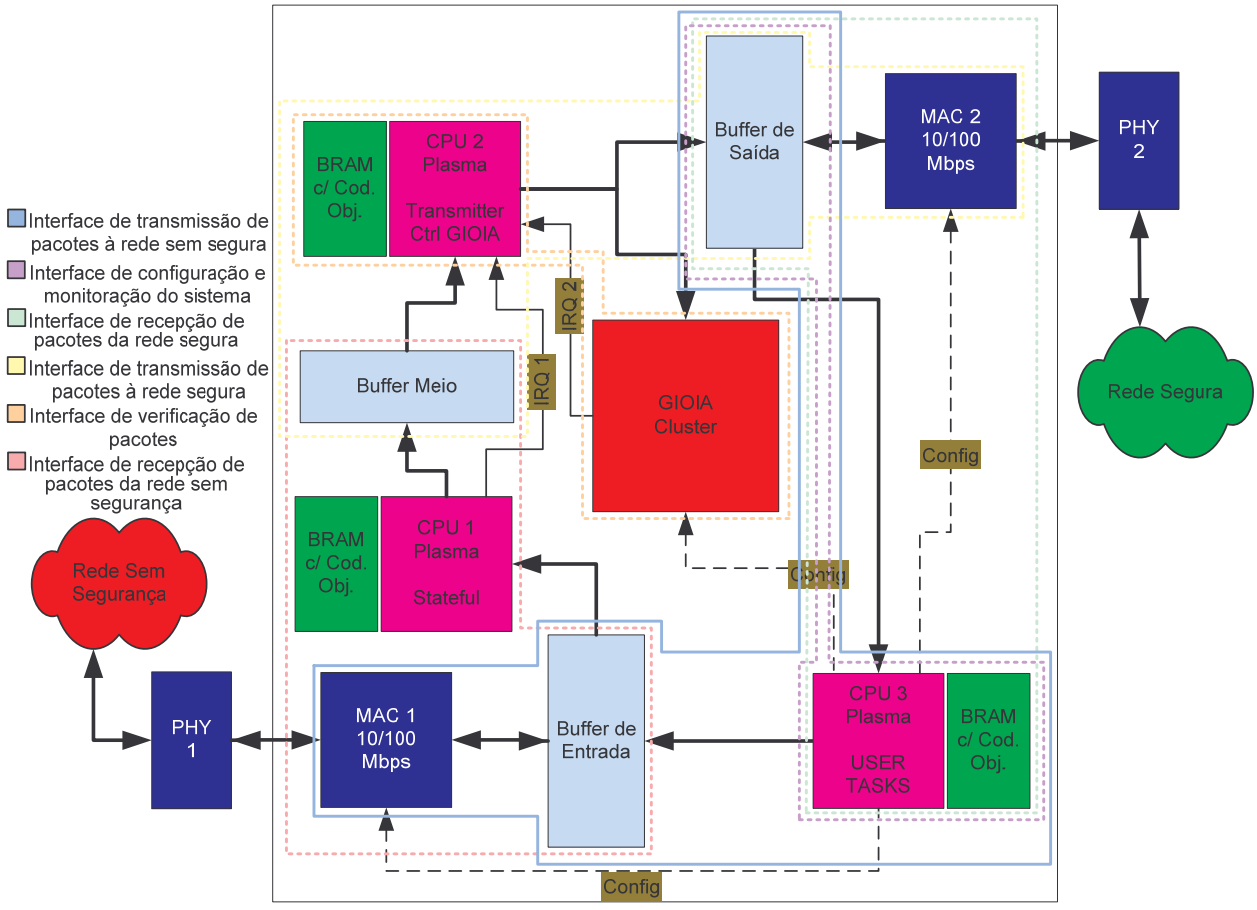


Figura 27 - Principais interfaces do sistema PrimeSec.

5.5.1 A interface de recepção de pacotes recebidos da rede sem segurança

Esta interface é responsável pela recepção e armazenamento dos pacotes da rede não segura, ela é formada por quatro módulos: o MAC Ethernet 1, o Buffer de Entrada, o processador Plasma 1 e o Buffer Meio. O fluxo dos dados de um pacote nesta interface é detalhado abaixo e descrito na

Figura 28.

O MAC Ethernet 1 é responsável pela recepção dos pacotes Ethernet na camada física padrão Ethernet (o PHY) nibble à nibble, e por inserí-los corretamente no Buffer de Entrada (ação 1, na Figura 28). Este Buffer de Entrada possui duas máquinas de estados que realizam a tradução do protocolo de handshake do MAC de recepção e transmissão dos bytes do pacote em escritas em memória. O Buffer de Entrada possui uma estrutura de memória dividida em quatro páginas de 2 kilobytes cada. Este tamanho foi escolhido para que pudéssemos manter um pacote Ethernet por página. Estas quatro páginas foram implementadas em memórias RAM dupla porta presentes no FPGA, possibilitando que sejam realizadas, simultaneamente, operações de escrita (pelo MAC 1) e operações de leitura (pelo Plasma 1) no Buffer de Entrada. A estrutura de páginas foi necessária para realizar uma adaptação entre as diferentes velocidades de escrita pelo MAC 1 e leitura pelo processador.

Ao receber um pacote inteiro, a máquina de estados presente no Buffer de Entrada envia uma interrupção ao processador Plasma 1 (ação 2). O software do processador copia o pacote da memória, armazenando-o no Buffer Meio (ação 3). Este buffer possui a mesma estrutura de páginas de pacotes presentes no Buffer de Entrada, porém não possui as máquinas de estados presentes no mesmo.

Após realizar a cópia, o processador Plasma 1 sinaliza a existência de um novo pacote ao Plasma 2, através de uma interrupção (ação 4). Este então começa a realizar o tratamento do pacote.

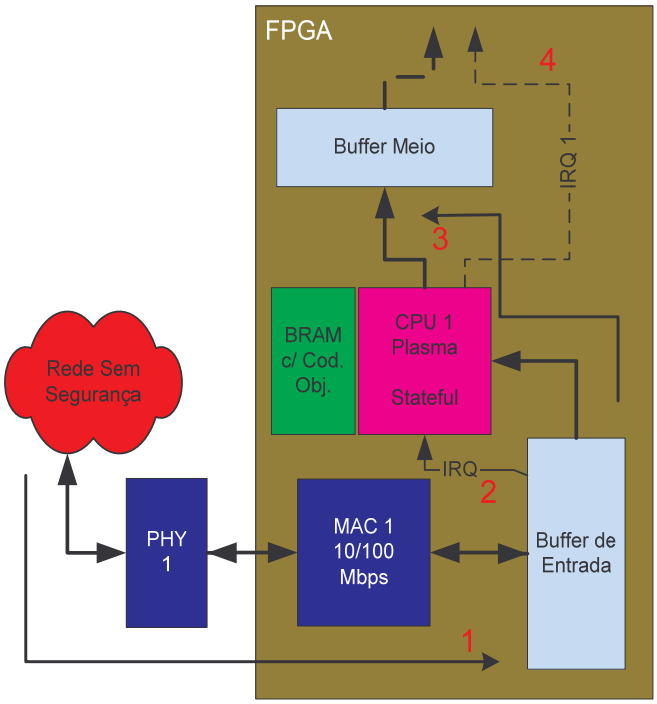


Figura 28 - Fluxo de dados de um pacote recebido no sistema.

Na proposta inicial do presente Trabalho de Conclusão, o processador Plasma 1 executaria algoritmos de remontagem de pacotes fragmentados (*reassembly*) e o Buffer Meio seria uma estrutura SRAM externa ao FPGA. Porém, não foi possível realizar tais implementações devido ao curto período de tempo para a realização do trabalho. Optou-se em manter este processador no sistema para permitir a implementação destes algoritmos em trabalhos futuros. No entanto, sem estes algoritmos os pacotes fragmentados serão analisados individualmente, o que torna o sistema

vulnerável à ataques por fragmentação de pacotes.

5.5.2 A interface de verificação de pacotes

Esta interface é responsável pela verificação dos pacotes já armazenados em memória (Buffer meio). Ela é formada por dois módulos: o processador Plasma 2 e o cluster GIOIA operando como co-processador de detecção de padrões. O fluxo de dados de um pacote nesta interface é detalhado abaixo e descrito na Figura 29.

Após receber uma interrupção do processador Plasma 1, o Plasma 2 inicia o tratamento deste pacote. Primeiramente, o software do Plasma 2 realiza uma verificação da operabilidade do sistema, isto é, ele verifica se sinal ***sys_on*** está ativo (a geração deste sinal é posteriormente abordada na Seção 5.5.5, ação 1 na Figura 29), que informa se o cluster foi devidamente configurado. Desta forma, o software evita enviar dados ao cluster desnecessariamente.

Caso o sistema não esteja ativo, o pacote é repassado à interface de transmissão de pacotes à rede segura sem qualquer verificação do conteúdo (ação 5). Esta ação é necessária para que o sistema, se mal configurado, não afete o funcionamento da rede.

Caso o sistema esteja ativo, o software irá processar os cabeçalhos do pacote, e retirar a informação de qual é o protocolo utilizado (ação 2). Ao final desta operação, o Plasma 2 envia ao cluster a informação de protocolo do pacote e começa a transmissão do conteúdo da carga útil do pacote (ação 3).

Se o cluster detectar um ataque no conteúdo de um pacote em análise, ele irá sinalizar ao processador Plasma 2 através de uma interrupção (ação 4). Ao receber uma interrupção do cluster, o Plasma 2 realiza um cancelamento no envio deste pacote pela interface seguinte, finaliza o envio de bytes ao cluster, salva a informação de qual regra foi detectada, monta um pacote de monitoração com esta informação e o envia para a interface de transmissão de pacotes.

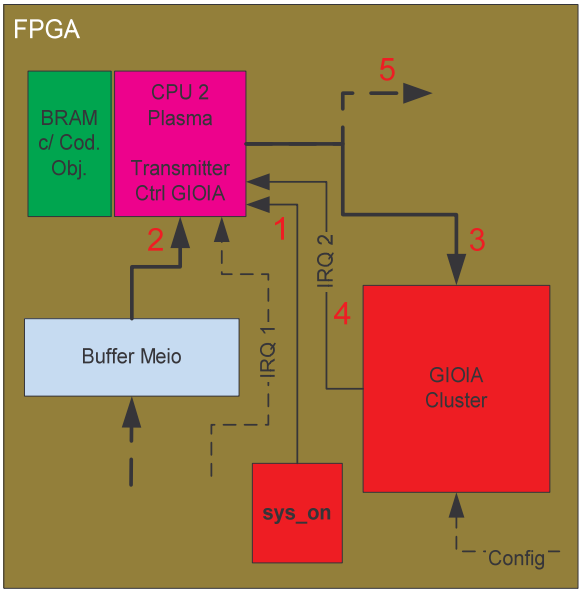


Figura 29 – Fluxo de dados de um pacote na interface de verificação de pacotes.

5.5.3 A interface de transmissão de pacotes à rede segura

Esta interface é responsável pelo envio dos pacotes já armazenados em memória para a rede segura, ela é formada por quatro módulos: o Buffer Meio, o processador Plasma 2, o Buffer de

Saída e o MAC Ethernet 2. O fluxo de dados de um pacote nesta interface é detalhado abaixo e descrito na Figura 30.

Logo após a verificação da operabilidade do sistema (ações 1 e 2, na Figura 30), já descrito na Seção 5.5.2, o processador Plasma 2 envia os bytes referentes aos cabeçalhos do pacote para uma nova memória, o Buffer de Saída (ação 3). A estrutura do Buffer de Saída é a mesma utilizada no Buffer de Entrada, com duas máquinas de estados tanto para recepção quanto para transmissão dos bytes ao MAC Ethernet.

Se o sinal ***sys_on*** não estiver habilitado o processador envia todo o conteúdo do Buffer Meio para o Buffer de Saída, e então dispara a máquina de transmissão presente neste módulo. Caso o sinal ***sys_on*** esteja habilitado, o Plasma 2 realiza o envio de bytes ao cluster para serem processados, e também os envia para o Buffer de Saída para transmissão (ação 3). A transmissão de bytes é feita em palavras de 32 bits. Caso o processador receba uma interrupção do cluster, informando a detecção de um ataque no pacote que está sendo processado, ele ativa um registrador de cancelamento de pacote presente na estrutura do Buffer de Saída e a transmissão deste pacote é abortada.

A máquina de estados de transmissão do Buffer de Saída, ao ser disparada interage com o MAC Ethernet para realizar a transmissão do pacote armazenado em sua memória. Esta máquina de estados traduz as operações de leitura da memória em um protocolo de handshake utilizado no MAC Ethenet. Esta máquina de estados pode cancelar a transmissão do pacote em transmissão, sinalizando o MAC um erro de transmissão através do sinal ***error_in_pc***. O MAC Ethernet 2 realiza o envio do pacote à camada física PHY, nibble à nibble.

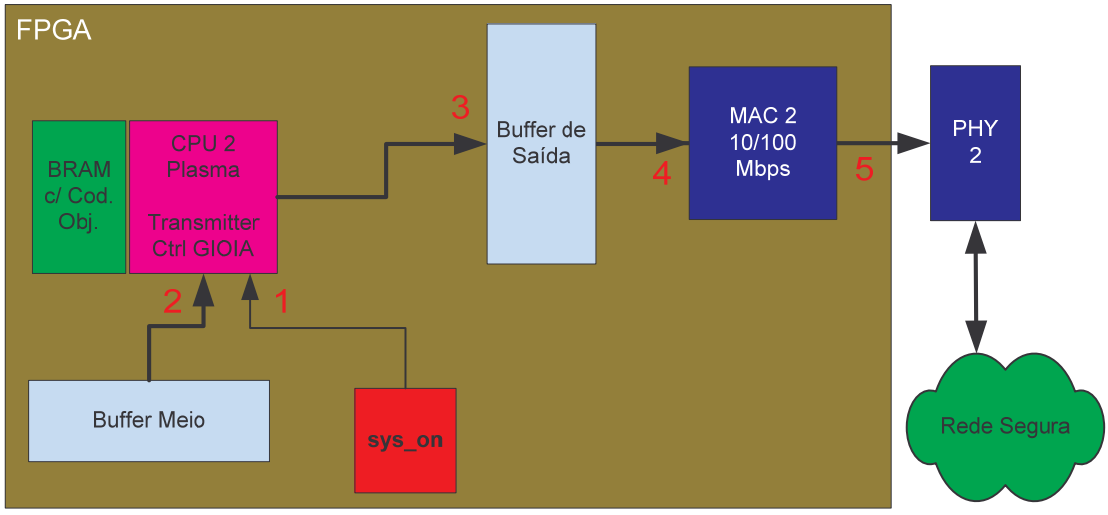


Figura 30 - Fluxo de dados de um pacote na interface de transmissão de pacotes à rede segura.

5.5.4 A interface de recepção de pacotes da rede segura

Esta interface é responsável pela recepção e armazenamento dos pacotes da rede segura. Ela é formada por três módulos: o MAC Ethernet 2, o Buffer de Saída, o processador Plasma 3. O fluxo de dados de um pacote nesta interface é detalhado abaixo e descrito na Figura 31.

A estrutura desta interface é similar à estrutura da interface de recepção de pacotes da rede não segura. O MAC Ethernet 2 recebe os pacotes através da camada física PHY, nibble à nibble (ação 1, na Figura 31), e os repassa byte à byte à máquina de estados de recepção presente no

Buffer de Saída (ação 2).

O Buffer da Saída envia uma interrupção ao Plasma 3 a cada duzentos bytes recebidos. Esta ação acontece porque o Plasma 3 necessita de tempo para processar os cabeçalhos do pacote armazenado em memória, buscando pacotes de configuração/monitoração do sistema. Estes pacotes de configuração/monitoração do sistema são dirigidos ao próprio sistema, não sendo repassados à rede sem segurança. Ao receber esta interrupção, o processador Plasma 3 imediatamente inicia o tratamento do pacote. Ao receber a informação de que todo o pacote foi armazenado em memória, a máquina de estados gera uma nova interrupção ao Plasma 3, sinalizando o fim da recepção do pacote.

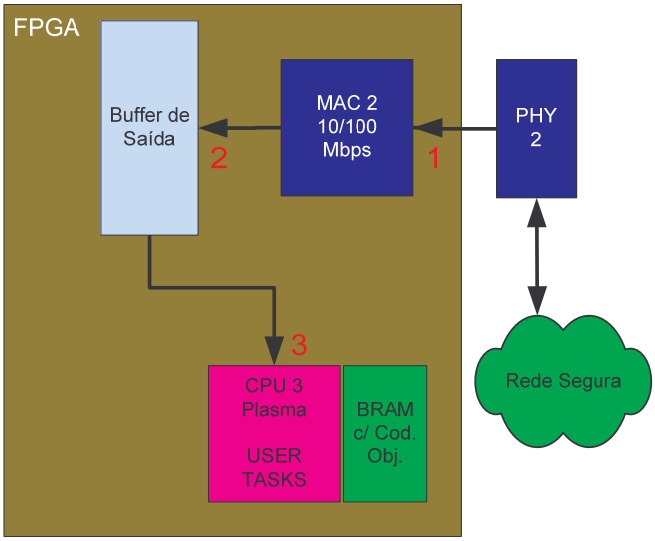


Figura 31 - Fluxo de dados de um pacote na interface pacotes da rede segura.

5.5.5 A interface de configuração e monitoração do sistema

Esta interface é responsável pelo processamento dos pacotes recebidos na interface anterior, buscando pacotes de configuração/monitoração do sistema PrimeSec. A interface é formada por dois módulos: o Buffer de Saída e o processador Plasma 3. O fluxo de dados de um pacote nesta interface é detalhado abaixo e descrito na Figura 32.

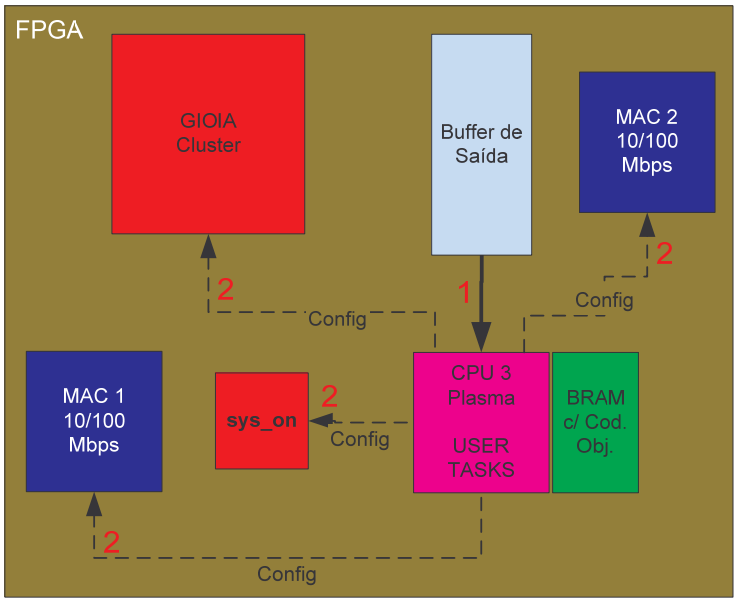


Figura 32 - Fluxo de dados de um pacote na interface configuração e monitoração do sistema.

Esta interface é responsável pelo tratamento dos pacotes de configuração do sistema. Estes pacotes são construídos sobre o protocolo IP e possuem uma estrutura de cabeçalho descrita na Figura 33.

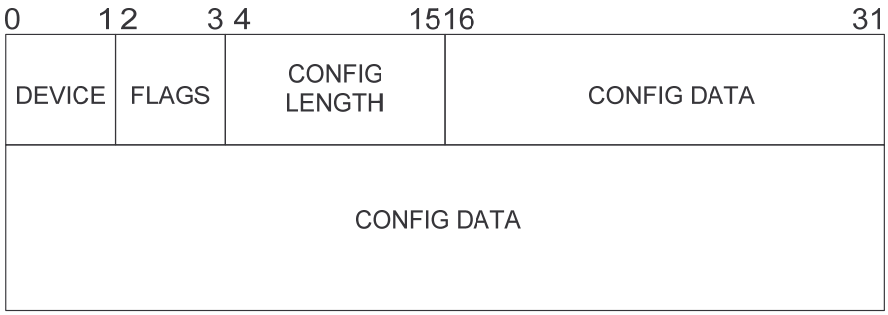


Figura 33 - Estrutura do cabeçalho de um pacote de configuração.

Os campos do cabeçalho de um pacote de configuração são os seguintes:

- **Dispositivo (do inglês, *device*):** Utilizado para informar qual módulo deverá ser configurado: GIOIA, MAC1, MAC2 ou SISTEMA;
- **Flags:** Somente utilizado quando o campo **dispositivo** estiver com o valor de SISTEMA, o campo flags informa qual ação deverá ser tomada: ativar ou desativar o sistema;
- **Tamanho da configuração (do inglês, *configuration length* ou *config length*):** Utilizado para informar o tamanho, em bytes, do campo de dados no pacote de configuração;
- **Dados de configuração (do inglês, *configuration data* ou *config data*):** Este campo possui os dados de configuração a serem escritos no módulo descrito pelo campo **dispositivo**.

Ao detectar um pacote de configuração/monitoração (ação 1), o software presente no Plasma 3 realiza uma das seguintes ações:

- Configuração do cluster GIOIA: Ao receber um pacote de configuração de regras, o processador transmite ao cluster GIOIA as regras presentes na carga útil do pacote.
- Configuração do MAC Ethernet: O banco de registradores de qualquer um dos MACs pode ser configurado através de um pacote de configuração dos MACs. A comunicação com a interface MII também é realizada através destes pacotes. Ao receber um pacote deste tipo, o processador faz uma escrita no banco de registradores do MAC Ethernet apontado pelo pacote de configuração;
- Inicialização/Parada do sistema: O sistema pode ser inicializado ou parado através de um pacote start/stop. Ao receber um pacote de configuração deste tipo, o processador habilita/desabilita o sinal ***sys_on***, o qual informa a interface de verificação de pacotes que o cluster já foi devidamente configurado e está pronto para receber regras. O sistema não garante que o cluster já tenha sido configurado antes de receber este pacote de inicialização. Esta tarefa cabe ao sistema de monitoração do PrimeSec.

Ao término do tratamento do pacote de configuração, este é imediatamente descartado pelo Plasma 3 não sendo repassado à interface de transmissão à rede não segura. Caso um pacote de configuração/monitoração não seja detectado no processamento dos cabeçalhos do pacote em memória, este é repassado para a interface de transmissão à rede não segura.

5.5.6 A interface de transmissão de pacotes à rede sem segurança

Esta interface é responsável pela transmissão dos pacotes já armazenados em memória à rede não segura. Esta é formada por quatro módulos: o Buffer de Saída, o processador Plasma 3, o Buffer de Entrada e o MAC Ethernet 2. O fluxo de dados de um pacote nesta interface é detalhado abaixo e descrito na Figura 34.

Ao final do processamento dos cabeçalhos do pacote (ação 1), caso este não seja um pacote de configuração deverá ser transmitido à rede não segura. O software presente no processador Plasma 3 realiza esta operação enviando os duzentos bytes recebidos por interrupção à memória presente no Buffer de Entrada (ação 2), fazendo assim uma troca de memória. Esta operação se repete a cada duzentos bytes até o final do pacote. Ao final da primeira troca de memória, o processador dispara a máquina de estados de transmissão presente na estrutura do Buffer de Entrada. Esta máquina de estados inicia a transmissão dos bytes do pacote ao MAC Ethernet 1 (ação 3), e este envia o pacote nibble à nibble à camada física PHY (ação 4).

Note que nenhuma verificação de regras ocorre no fluxo de pacotes da rede segura para a não segura, sendo assim o cluster GIOIA só irá atuar no conjunto de bytes dos pacotes de entrada da rede não segura.

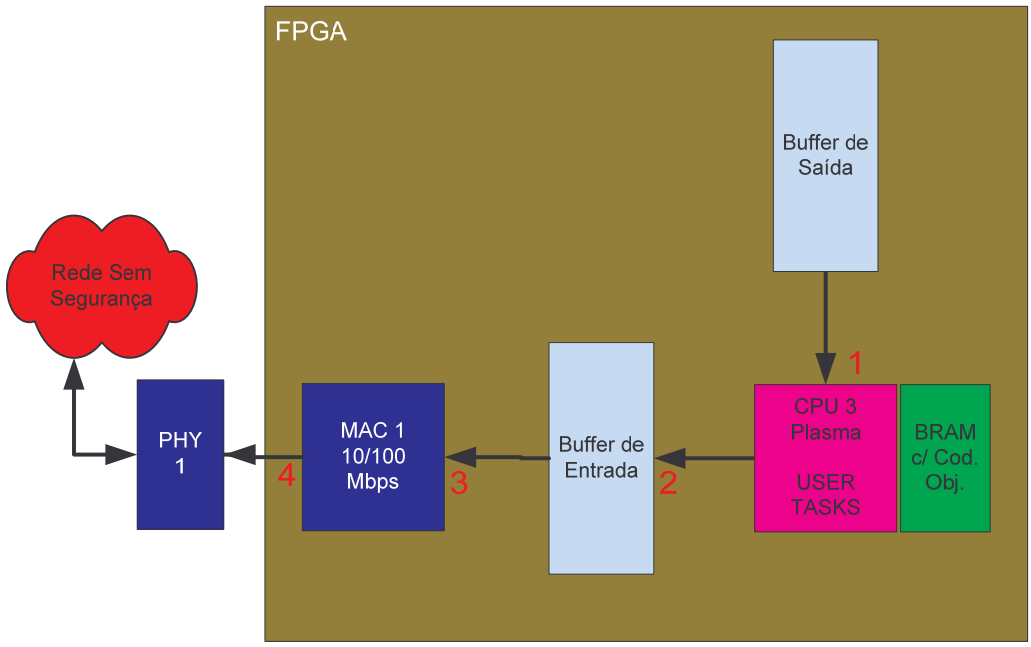


Figura 34 - Fluxo de dados de um pacote na interface transmissão de pacotes à rede sem segurança

6 Estratégia de Validação Funcional do PrimeSec

Neste Capítulo é abordada a validação funcional do sistema PrimeSec, apresentando-se inicialmente a estrutura de testbench utilizada na simulação do sistema, e a seguir descrevendo a validação funcional de cada uma das seis interfaces descritas na Seção 5.5.

6.1 Estrutura do testbench utilizada

A validação funcional é iniciada criando-se um modelo, em VHDL, para a camada física padrão Ethernet (PHY). Este modelo funcional de um PHY simula as condições presentes em uma rede física operando a 100 Mbps, no modo *full-duplex*. Este testbench lê os bytes a serem transferidos ao MAC em um arquivo chamado *fromPHY.txt*, este arquivo possui pacotes simulados que serão injetados nos MACs Ethernet 10/100 Mbps presentes no sistema. A Figura 35 exibe uma visão geral da estrutura do testbench desenvolvido para a validação funcional do sistema.

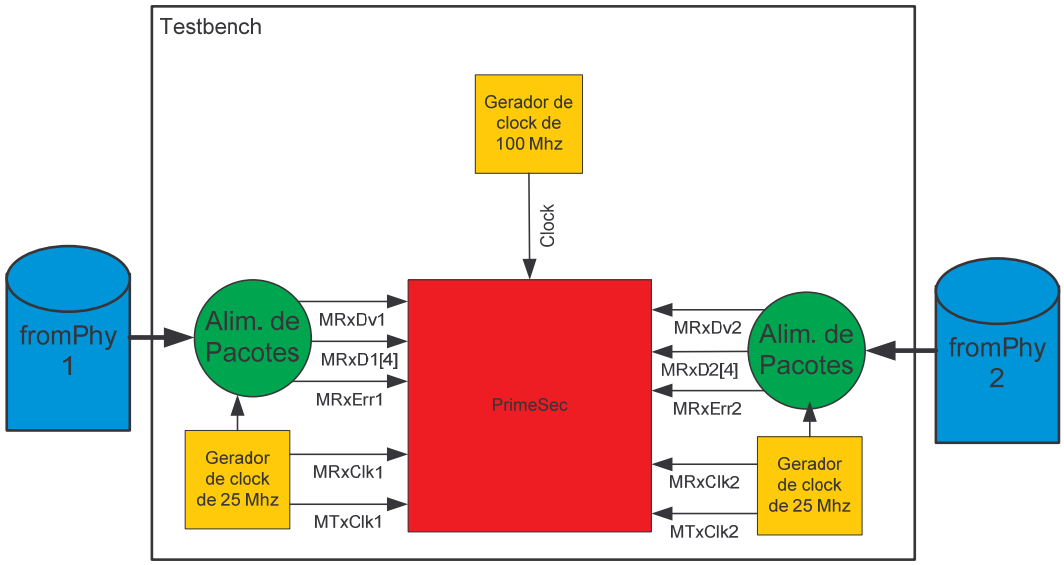


Figura 35 - Estrutura do testbench utilizado nas simulações do sistema PrimeSec.

Os Alimentadores de Pacotes realizam o papel dos PHYs conectados aos MACs, a partir do arquivo de pacotes simulados *fromPHY.txt* correspondente ao PHY em simulação. O arquivo *fromPHY1.txt* contém pacotes com ataques em meio a pacotes normais. O arquivo *fromPHY2.txt* contém pacotes de configuração de regras com destino à rede sem segurança em meio a pacotes normais.

A geração dos pacotes de ataques e das de regras é realizada por scripts, desenvolvidos para converter as regras do Snort e seus ataques ao formato de pacotes utilizado nos arquivos *fromPHY.txt*.

6.2 Validação da interface de recepção de pacotes da rede sem segurança

Ao iniciar a simulação do sistema, o alimentador de pacotes inicia a sua função e emula um PHY Ethernet, repassando nibble à nibble os bytes do pacote simulado presentes no arquivo *fromPHY1.txt* ao MAC Ethernet 1. O MAC 1, ao receber o sinal ***MRxDv***, inicia a recepção dos dados de um pacote, presentes no barramento ***MRxD.*** Caso o MAC1 receba um sinal de erro na recepção dos pacotes pelo PHY, através do sinal ***MRxErr***, ele deverá abortar a recepção do pacote. O MAC então repassa o pacote ao Buffer de Entrada, através dos sinais ***active_out***, ***tx*** e do

barramento ***data_out*** (8 bits). Os sinais de inteface entre os módulos são apresentados na Figura 25, página 31. A Figura 36 exemplifica uma recepção de pacote realizada pelo MAC Ethernet 1.

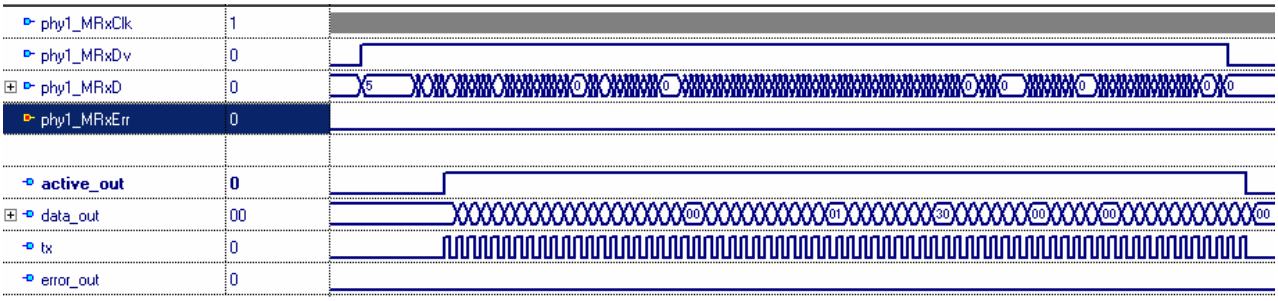


Figura 36 - Recepção de um pacote no MAC Ethernet 1.

Ao receber um pacote completo em sua memória, o Buffer de Entrada interrompe o processador Plasma 1 através do sinal ***rcvd_total*** ligado ao barramento de propósito geral ***p1_gpioA_in***. O barramento ***p1_gpioA_in*** está conectado ao vetor de interrupções do Plasma. Este então troca o pacote de memória através de seus barramentos de acesso à memória externa ***p1_mem_write***, ***p1_mem_address***, ***p1_mem_data*** e ***p1_mem_byte_sel***. A Figura 38 exemplifica a interrupção gerada pelo Buffer de Entrada e a troca da memória realizada pelo Plasma 1.

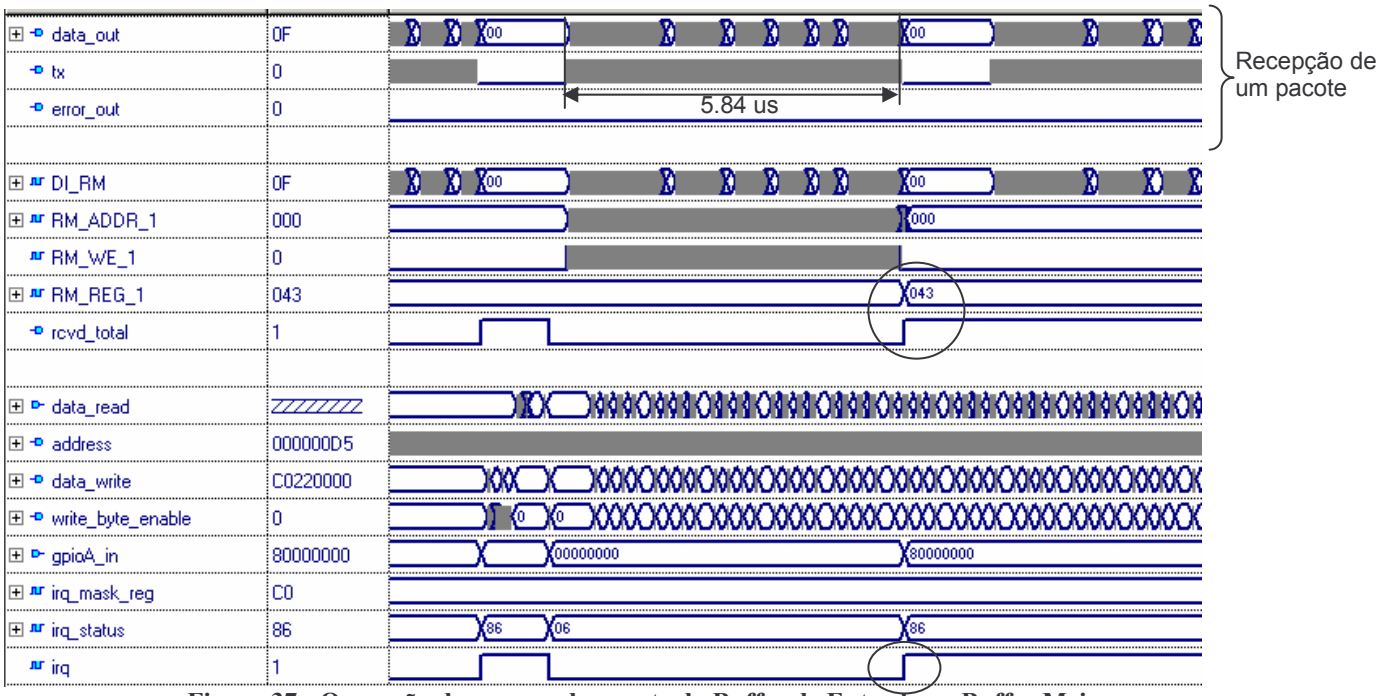


Figura 37 - Operação de repasse do pacote do Buffer de Entrada ao Buffer Meio.

Na simulação acima é apresentada a recepção de um pacote de 67 bytes de carga útil (RM_REG_1=43H), transmitidos em 5.84 us. Adicionado-se o número de bytes correspondente ao preâmbulo e CRC, obtém-se uma taxa de transmissão de 100 Mbps.

Ao terminar a troca de buffers, o processador Plasma 1 envia uma interrupção ao processador Plasma 2. Esta ação é realizada através de uma escrita no registrador ***changed***, presente na estrutura do Buffer Meio. A Figura 39 exemplifica esta escrita no registrador ***changed***, a interrupção sendo gerada no processador Plasma 2, e o tempo que leva a operação de repasse deste pacote.

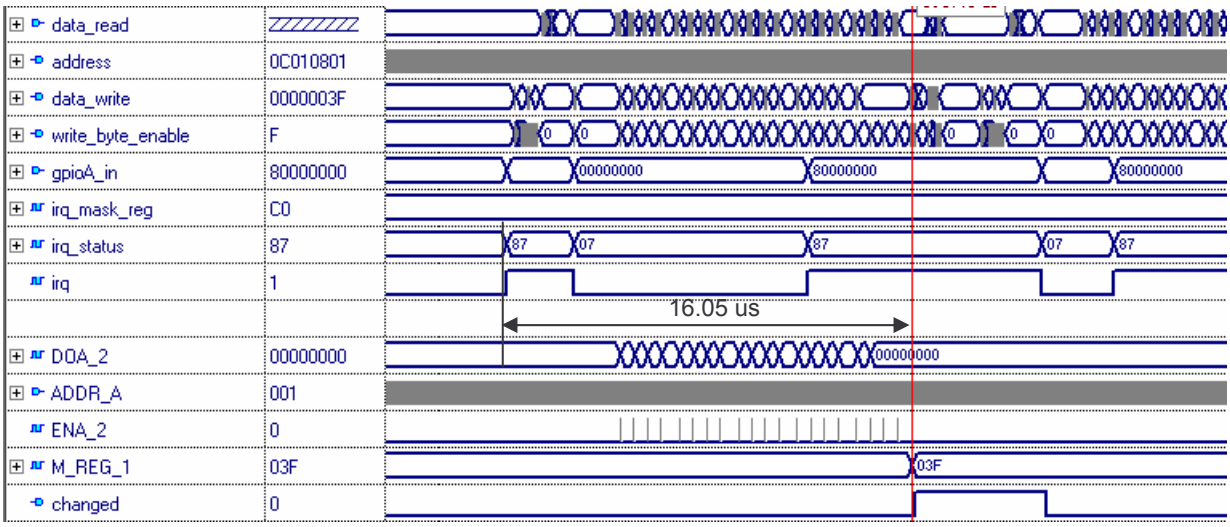


Figura 38 - Pacote sendo escrito no Buffer Meio e a interrupção gerada ao Plasma 2.

6.3 Validação da interface de verificação dos pacotes

Ao receber a interrupção, através do sinal ***changed***, o processador Plasma 2 verifica o sinal ***sys_on*** para determinar se deve enviar o pacote ao cluster GIOIA, ou não. Em caso afirmativo, o Plasma 2 realiza a verificação dos cabeçalhos do pacote e determina o protocolo deste. Ao final desta ação, o software envia o protocolo para o cluster (os sinais do cluster que participam desta ação são: ***pakdat***, ***isproto*** e ***take***) e em seguida envia palavras de 32 bits (quatro bytes) tanto ao cluster para serem comparados, quanto ao Buffer de Saída para serem transmitidos. O cluster possui uma estrutura de wrapper que recebe estas palavras de quatro bytes e os repassa ao cluster byte à byte, através dos sinais ***pakdat*** e ***take***. A Figura 39 exemplifica o tratamento do pacote pelo Plasma 2 e as escritas no cluster.

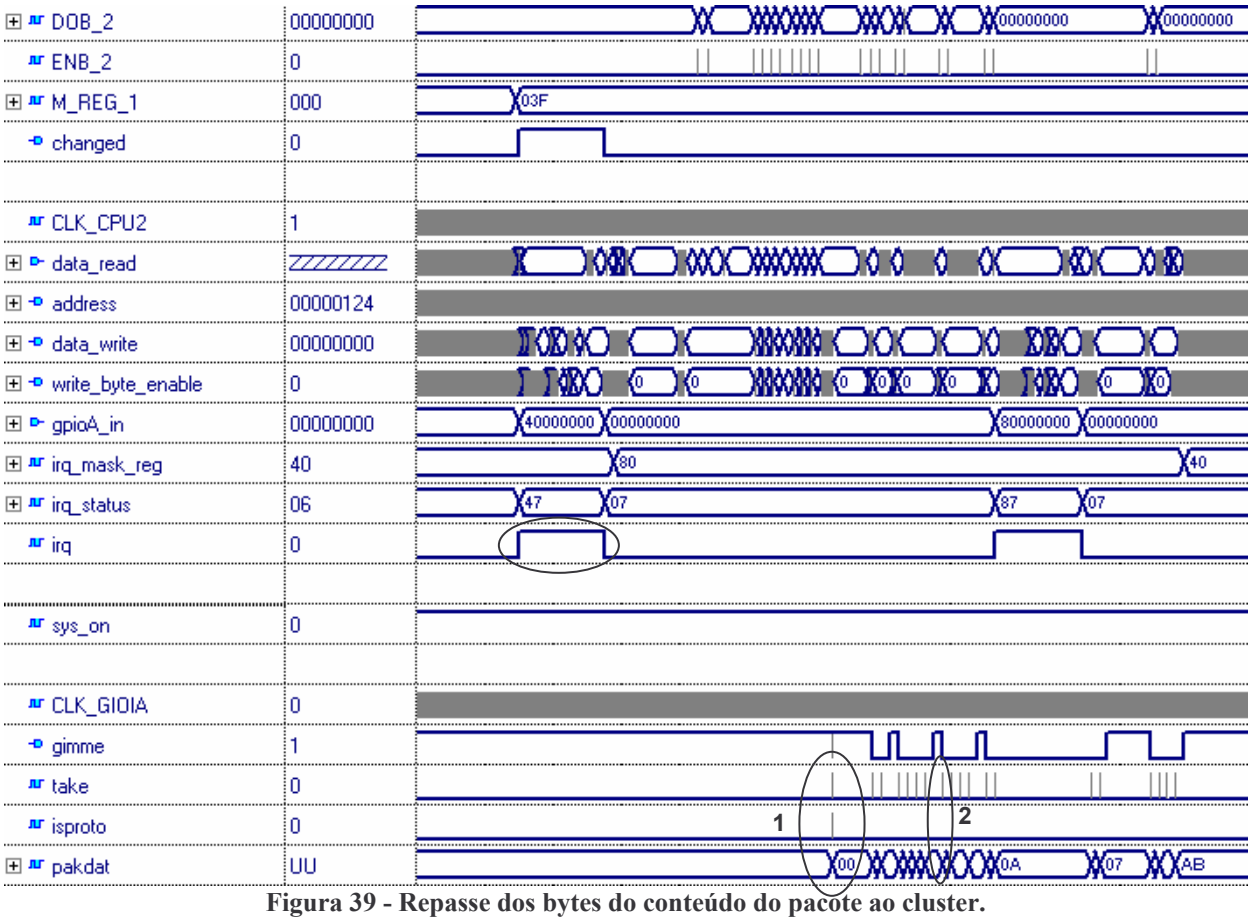


Figura 39 - Repasse dos bytes do conteúdo do pacote ao cluster.

Os pontos **1** e **2,** destacados na Figura 39, correspondem à velocidade na qual o cluster está recebendo bytes, visto que o sinal *take* indica a disponibilidade de um novo dado no barramento pakdat. Como apresentado previamente, o tempo padrão para solicitar um byte é proporcional ao número de regras contidas na pico-CPU multiplicado pelo período de relógio. Nesta simulação, o número de regras é igual a 40, e o período aplicado ao cluster igual a 5 ns, resultando em um tempo de byte igual a 200 ns. O tempo mostrado na simulação, 0,22 µs, corresponde a este tempo. Se limitarmos o número de regras por pico-CPU em 16, mantendo a freqüência de operação em 200 MHz, é possível tratar em tempo real taxas de 100 Mbps (tempo de byte igual a 80 ns).

Caso o cluster detecte uma regra no pacote em comparação, ele irá gerar uma interrupção para o Plasma 2 sinalizando-o deste fato. O Plasma 2 imediatamente lê do cluster a informação de qual regra foi detectada. A seguir, o software cessa o envio dos bytes ao cluster e ao Buffer de Saída, não sinalizando a transmissão deste pacote à máquina de estados de transmissão e assim realizando o descarte deste pacote. A Figura 40 exemplifica a interrupção gerada pelo cluster ao Plasma 2, quando este detecta um ataque no conteúdo do pacote.

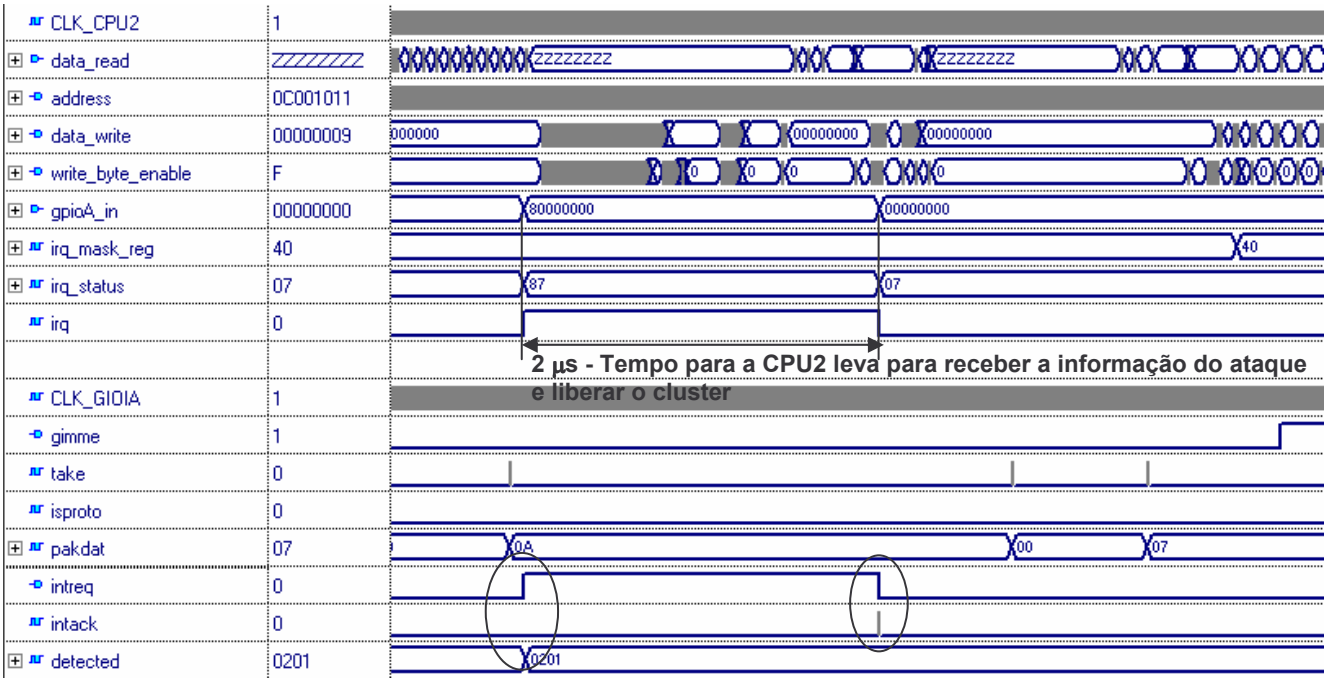


Figura 40 - Sinalização da detecção de uma regra no pacote em comparação via interrupção.

6.4 Validação da interface de transmissão de pacotes à rede segura

O processador Plasma 2, ao mesmo tempo que envia os bytes do pacote ao cluster, os envia ao Buffer de Saída para a transmissão à rede segura. Ao final desta operação, se não ocorrer nenhuma interrupção que indique uma detecção de ataque contido no pacote, o Plasma 2 dispara a máquina de estados de transmissão presente na estrutura do Buffer de Saída. A Figura 41 exemplifica as escritas na memória do Buffer de Saída e o disparo da máquina de transmissão presente no módulo.

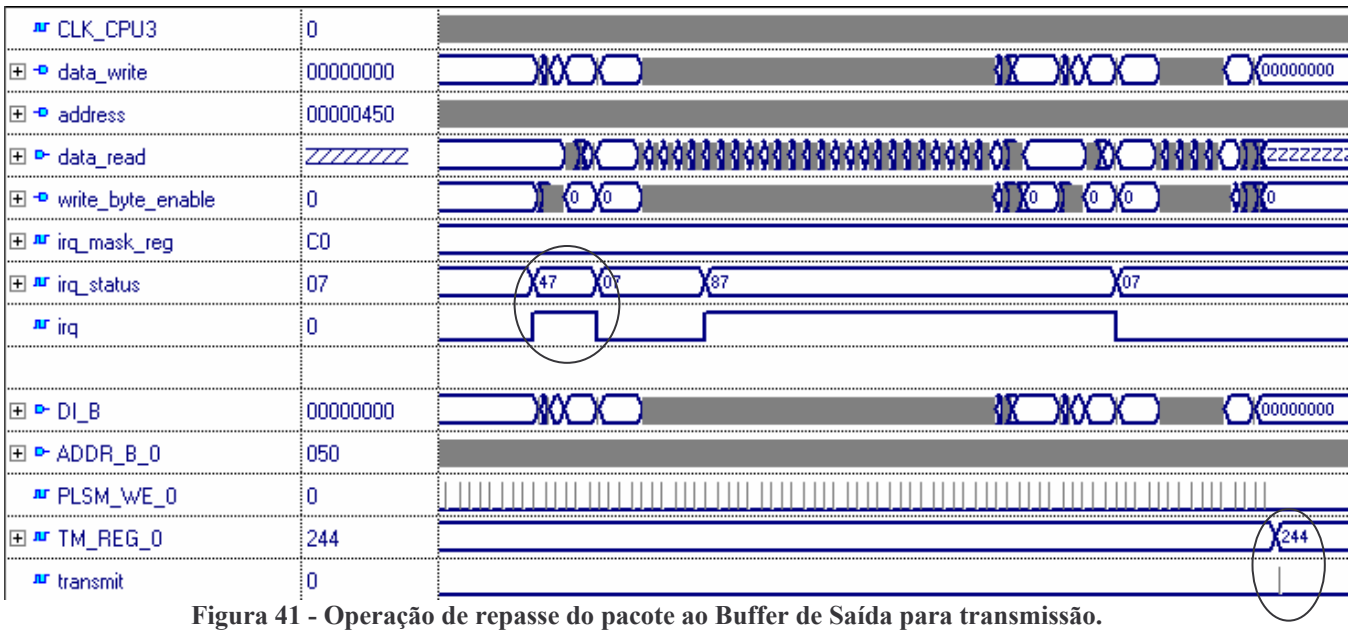


Figura 41 - Operação de repasse do pacote ao Buffer de Saída para transmissão.

Ao ser disparada, a máquina de estados de transmissão do Buffer de Saída inicia-se a transmissão do pacote, byte à byte, ao MAC 2 que os envia ao PHY2, nibble à nibble. A Figura 42 exemplifica a transmissão de um pacote pelo MAC2 ao PHY2, e consequentemente, à rede segura.

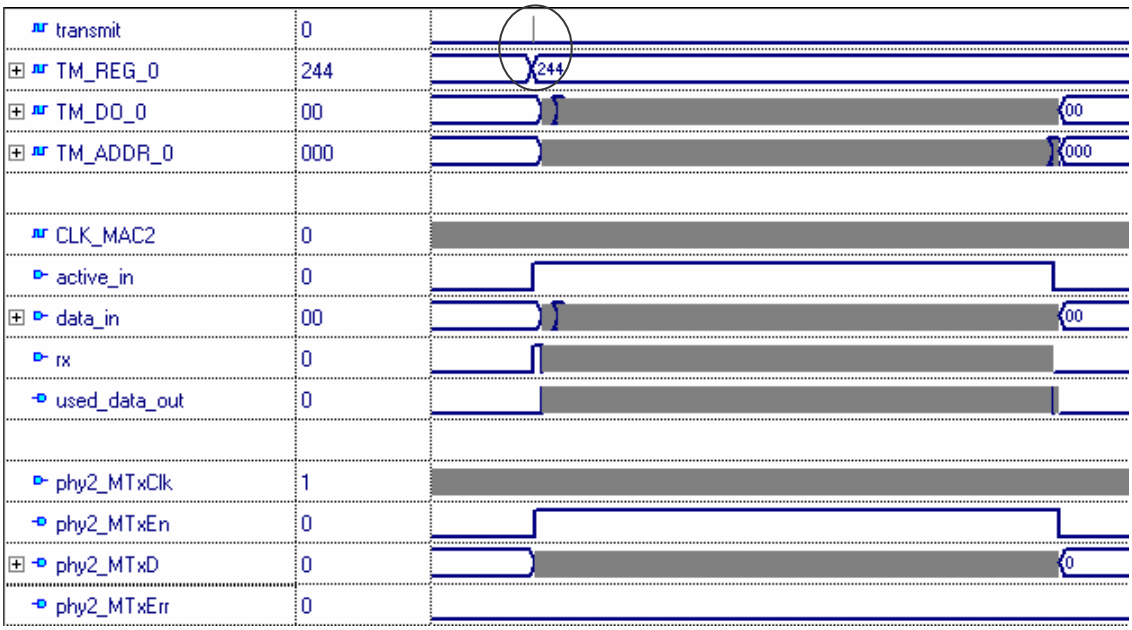


Figura 42 - Transmissão de um pacote do MAC Ethernet 2 à rede segura.

6.5 Validação da interface de recepção de pacotes da rede segura

Da mesma forma que o MAC Ethernet 1 realiza a recepção dos pacotes da rede sem segurança, o MAC Ethernet 2 realiza a recepção dos pacotes da rede segura. A Figura 43 exemplifica a recepção de um pacote pelo MAC 2 e seu repasse ao Buffer de Saída.

Figura 43 - Recepção de um pacote no MAC Ethernet 2.

A principal diferença desta interface para a interface de recepção da rede sem segurança, é que nesta interface a máquina de estados de recepção presente na estrutura do Buffer de Saída não espera o armazenamento de um pacote inteiro em sua memória para interromper o processador Plasma 3. A interrupção ocorre a cada duzentos bytes e é sinalizada através do sinal ***rcvd_pieces***, e ao receber o pacote completo realiza a interrupção através do sinal ***rcvd_total*** ambos conectados ao barramento ***p3_gpioA_in***. Este valor foi escolhido em um processo de tentativa e erro para se adequar a taxa de entrada e saída da rede. A Figura 44 exemplifica as interrupções geradas pelo Buffer de Saída ao Plasma 3.

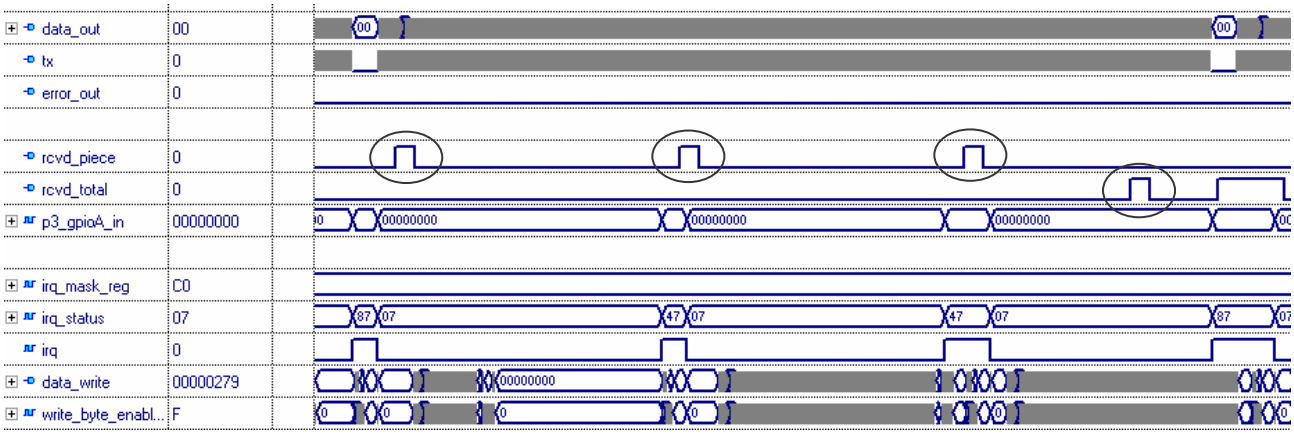


Figura 44 - Interrupção do Plasma 3 pelo Buffer de Saída.

6.6 Validação da interface de configuração e monitoração do sistema

Ao receber uma interrupção (via o sinal ***rcvd_pieces*** ou ***rcvd_total***), o processador Plasma 3 verifica se o pacote é de configuração, utilizando seu barramento de acesso à memória externa ***p3_mem_write***, ***p3_mem_address***, ***p3_mem_data*** e ***p3_mem_byte_sel***. Caso seja detectada a existência de dados de configuração, o Plasma 3 se encarrega de configurar o módulo para o qual o pacote de configuração foi criado, também através do barramento de acesso à memória externa. A Figura 45 exemplifica a configuração das regras presentes no cluster GIOIA.

Os últimos quatro sinais exibidos na simulação estão presentes no módulo GIOIA e são utilizados na configuração de regas do cluster (como descrito na Seção 4.1.4, pág.22).

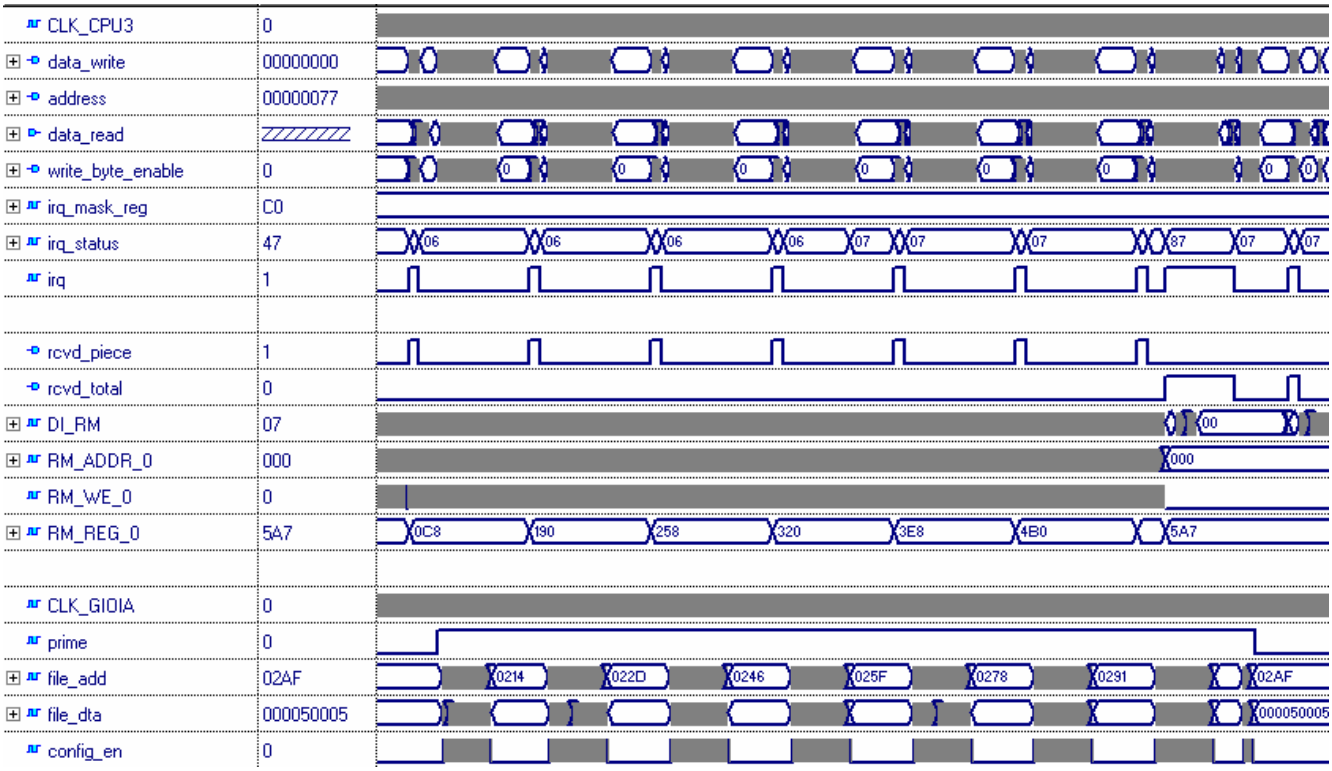


Figura 45 - Configuração das regras do cluster através da interface de configuração e monitoração do sistema.

6.7 Validação da interface de transmissão de pacotes a rede sem segurança

Após o processamento dos cabeçalhos do pacote armazenado no Buffer de Saída, ao não se detectar um pacote de configuração de dispositivos este deve ser transmitido à rede sem segurança. Esta operação é iniciada pela troca de memória realizada pelo processador Plasma 3, o pacote é transferido do Buffer de Saída para o Buffer de Entrada de onde será consumido. A Figura 46 exemplifica a troca de memória realizada pelo processador Plasma 3.

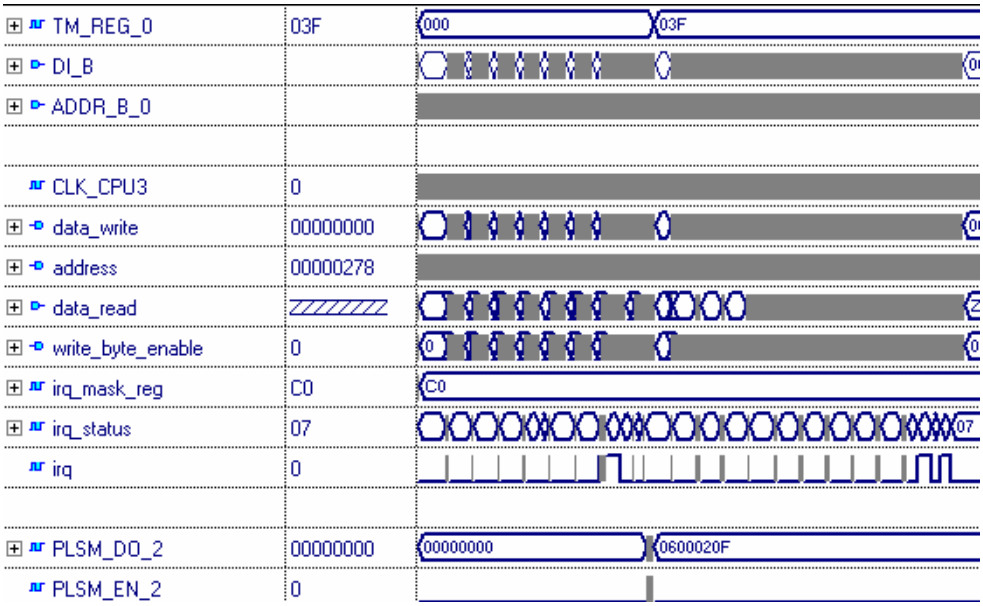


Figura 46 – Troca de memória realizada pelo Plasma 3.

Ao final da operação de troca de memória, o Plasma 3 dispara a máquina de estados de transmissão presente no Buffer de Saída via escrita no registrador ***transmit***. Ao ser disparada, a máquina de estados inicia a transação de dados com o MAC Ethernet 1 para enviar o pacote

armazenado em memória. Esta operação é similar a descrita na Seção , e pode ser observada na Figura 47.

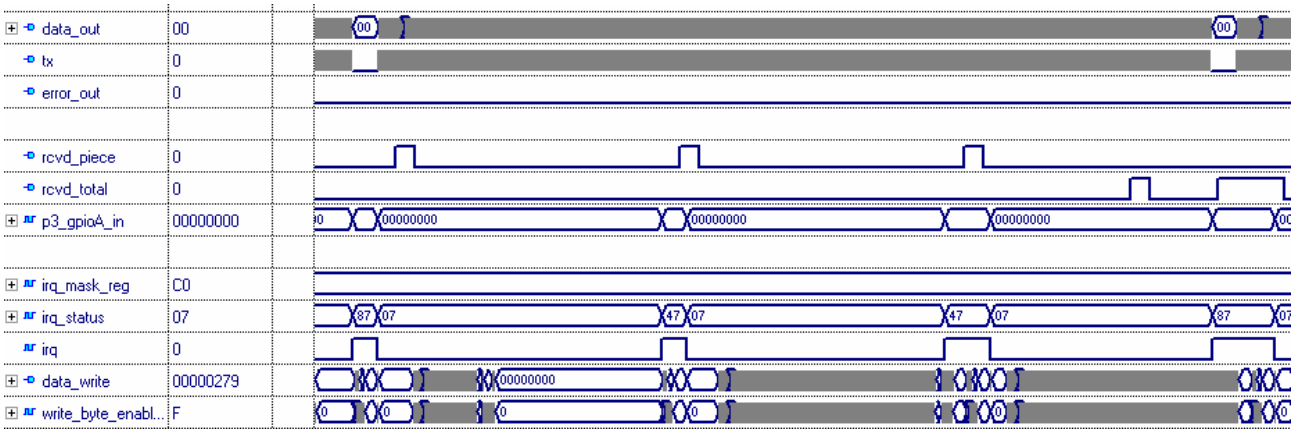


Figura 47 - Interrupção do Plasma 3 pelo Buffer de Saída

7 Conclusões e Trabalhos Futuros

O presente Trabalho de Conclusão apresentou o desenvolvimento de um sistema multi-processado em um único circuito integrado (MPSoC – *multi processor system-on-a-chip*). Este trabalho teve por característica a multi-disciplinaridade, requerendo dos Autores a aplicação dos conteúdos de arquiteturas e organização de computadores, algoritmos de programação e redes de computadores abordados durante o curso de Engenharia de Computação.

O sistema desenvolvido, *PrimeSec*, adotou como método de projeto o reuso de módulos de propriedade intelectual, evitando-se assim o desenvolvimento de cada componente, permitindo a construção de um sistema de razoável complexidade. O trabalho envolveu o desenvolvimento de software para os processadores Plasma, bem como todas as interfaces entre os componentes. Destaca-se como uma das dificuldades encontradas ao longo do trabalho o desenvolvimento do software, pois o mesmo requer considerações sobre o tempo de recepção e transmissão dos pacotes. No que se refere ao desenvolvimento do hardware, as maiores dificuldades ocorreram na definição dos protocolos de comunicação (interrupções e mapeamentos de endereços) entre os vários elementos do *PrimeSec*.

Como características do *PrimeSec* destaca-se a partição das tarefas em diferentes processadores (Plasma), e a utilização de um *cluster* de processadores elementares (pico-CPU) operando em paralelo para detecção de ataques na rede. O sistema mostrou-se funcional, através de simulação VHDL. Resultados mostram que o sistema permite detectar ataques em tempo real, em redes operando em taxas de até 100 Mbps.

A prototipação em FPGA é o primeiro trabalho futuro a ser executado. A plataforma contendo o FPGA deve conter duas interfaces de rede, permitindo a efetiva comprovação da funcionalidade do *PrimeSec*. Uma vez o *PrimeSec* prototipado, tem-se disponível umZ "pré-produto", que poderá ser trabalhado para vir a ser um produto de fato.

Outros trabalhos a serem realizados são a inclusão da funcionalidade de remontagem de pacotes fragmentados (*reassembly*), a ser executada no processador Plasma 1, bem como a inserção de uma memória externa de alta capacidade, como exemplo uma SDRAM. Estas alterações permitirão ao sistema que opere com maior eficiência em situações reais de tráfego de rede.

8 Referências Bibliográficas

[ALL03] Allen, J. et al. *State of the Practice of Intrusion Detection Technologies*. Capturado em: http://www.sei.cmu.edu/pub/documents/99.reports/pdf/99tr028.pdf. Setembro 2003.

[BAK04] Backer, Z; Prassana, V. Automatic Synthesis of Efficient Intrusion Detection Systems on FPGAs. In: 14th International Conference on Field Programmable Logic and Applications (FPL), 2004, pp. 311-321.

[BAS04] Bastos, E. L.; Steffen, J. Análise das ferramentas de IDS Snort e PRELUDE quanto à eficácia na detecção de ataque e na proteção quanto à evasões. Revista Tecnologia e Tendências, v.3(1), Junho 2004, pp. 19-24.

[BER01] Bergamaschi, R.; et al. Automating the design of SOCs using cores. IEEE Design & Test of Computers, vol. 18(5), Setembro-Outubro. 2001, pp. 32-45.

[CAL04] Calazans, N.; Moraes, F.; Hessel, F. MR2 Multicycle Processor. Novembro, 2004. (In Portuguese). http://www.inf.pucrs.br/~calazans/ undergrad/orgcomp/arq_MR2.pdf.

[CAR05a] Caruso, L. "Proposal of a NIDS Architecture Accelerated by Hardware ". MSc Dissertation, FACIN-PPGCC-PUCRS, Março, 2005.

[CAR05b] Caruso, L.; Guindani, G.; Schmitt, H. SPP-NIDS – A Sea of Processors Platform for Network Intrusion Detection Systems. FACIN-PPGCC-PUCRS, Setembro, 2005.

[CAS03] Caswell, B.; Beale, J.; Foster, J. C. Snort 2.0 Intrusion Detection. Rockland: Syngress Publishing, 2003, 523p.

[COM97] Comer, Douglas E. Computer networks and internets. New Jersey: Prentice-Hall, Julho, 1997.

[DES03] Desay, N. Increasing Performance in High Speed NIDS. Capturado em: http://www.linuxsecurity.com/resource_files/intrusion_detection/Increasing_Performance_in_High_Speed_NIDS.pdf . Agosto, 2003.

[ETH06] IEEE 802.3 CSMA/CD (ETHERNET). Capturado em: http://www.ieee802.org/3/. Novembro, 2006.

[FOR06] Force10 Networks, Inc. P1 and P10: GbE Security Appliances. Capturado em: http://www.force10networks.com/products/p-series_overview.asp. Agosto, 2006.

[GUP97] Gupta, R.; Zorian, Y. Introducing Core-Based System Design. IEEE Design & Test of Computers, v.14(4), Outubro-Dezembro 1997, pp. 15-25.

[HAR97] Hartenstein, R.; Becker, J.; Herz, M.; Nageldinger, U. An Innovative Platform for Embedded System Design. In: Architekturen von Rechensystemen (ARCS), Dezembro1997, pp. 143-152.

[HAU97] Hauser, J.; Wawrzynek, J. Garp: a MIPS Processor with a Reconfigurable Coprocessor. In: IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), Outubro 1997, pp. 12-21.

[HER02] Herveille, R.; SoC Interconnection: Wishbone. Opencores - http://www.opencores.org/projects.cgi/web/wishbone/wishbone. Julho, 2002.

[KEM02] Kemmerer, R.; Vigna, G. Intrusion Detection: A Brief History and Overview, Security & Privacy. Computer, v.35(3), Abril 2002, 27-30.

[LAR05] Larus, J. “SPIM simulator”. Capturado em: http://www.cs.wisc.edu/~larus/spim.html. Setembro 2005.

[LAW02] Lawton, G. Open Source Security: Opportunity or Oxymoron? Computer, v.35(3), Março 2002, pp. 18-21.

[MOH06] Igor Mohor, Ethernet MAC 10/100 Mbps. Capturado em: http://www.opencores.org/projects.cgi/web/ethmac/overview. Agosto, 2006.

[NOR03] Norton, M; Roelker, D. Snort 2.0 Hi-Performance Multi-Rule Inspection Engine. Capturado em: http://www.sourcefire.com/technology/wp_request.html. Agosto, 2003.

[NOR03a] Norton, M.; Roelker, D. Snort 2.0 Rule Optimizer. Capturado em: http://www.sourcefire.com/technology/ wp_request.html. Agosto, 2003.

[PLM06] Rhoads, S. Plasma - most MIPS I(TM) opcodes. Capturado em: http://www.opencores.org/projects.cgi/web/mips/overview. Outubro, 2006.

[PTA03] Ptacek, T. H.; Newsham, T. N. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Capturado em: http://www.snort.org/docs/idspaper/. Agosto, 2003.

[SNO06] Snort. Capturado em: http://www.snort.org. Agosto, 2006.

[SPU00] Spurgeon, C. Ethernet: The Definitive Guide. O’Reilly and Associates, 2000.

[TCP03] Tcpdump.org. Tcpdump Manual. Capturado em: http://tcpdump.org/tcpdump_man.html.. Abril, 2003.

[TOR01] Torok, D. Projeto visando a prototipação do protocolo de acesso ao meio em redes ethernet. FACIN-PPGCC-PUCRS, Agosto, 2001.

[XIL05] Xilinx, Inc. Xilinx FPGA Product Tables - Virtex-II Platform FPGA. Capturado em: http://www.xilinx.com/products/tables/fpga.htm#sp2. Janeiro, 2005.

[XIL06] Xilinx Spartan-3 Complete Datasheet http://direct.xilinx.com/bvdocs/publications/ds099.pdf