MARCO ANTÔNIO FARIA BOTELHO

ALTA DISPONIBILIDADE EM FIREWALL UTILIZANDO PFSYNC E CARP SOBRE FREEBSD

Monografia de Pós-Graduação "Lato Sensu" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Prof. Msc. Sandro Melo

LAVRAS
MINAS GERAIS – BRASIL
2006

MARCO ANTÔNIO FARIA BOTELHO

ALTA DISPONIBILIDADE EM FIREWALL UTILIZANDO PFSYNC E CARP SOBRE FREEBSD

Monografia de Pós-Graduação "Lato Sensu" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

APROVADA em ____ de _____ de 2006.

_____

Prof. MSc. Joaquim Quinteiro Uchôa

_____

Prof. MSc. Denilson Vedoveto Martins

_____

Prof. MSc. Sandro Melo

(Orientador)

LAVRAS

MINAS GERAIS – BRASIL

Dedicatória

Á minha amada noiva, Flávia, pelo apoio, cumplicidade e amor que sempre me encorajaram a prosseguir nessa jornada.

Agradecimentos

Ao professor e orientador Sandro Melo pela paciência na orientação e incentivo que tornaram possível a conclusão desta monografia.

A todos os professores da ARL, que foram tão importantes no desenvolvimento desta pesquisa.

Sumário

Lista de Abreviaturas

PFSync - Packet Filter State Table Loggin Interface
CARP - Common Address Redundancy Protocol
UNIVALE - Universidade Vale do Rio Doce
BSD - Berkeley Software Distribution
GPL - General Public License
FSF - Free Software Foundation
TCP - Transmission Control Protocol
IP - Internet Protocol
OSI - Open System Interconnect
ISO - International Organization for Standardization
PF - Packet Filter
NAT - Network Address Translator
VRRP - Virtual Router Redundancy Protocol
FOSS - Free and Open-Source Software
MAC - Media Address Control
HSRP - Hot Standby Router Protocol
ARP - Address Resolution Protocol
CIT - Centro de Informática e Tecnologia
NIC - Network Interface Card
DMZ - Demilitarized Zone

Lista de Figuras

Lista de Tabelas

Resumo

Esta pesquisa apresenta uma solução para ambientes que exigem alta disponibilidade dos recursos computacionais através da redundância do *hardware* e *software – PFSync* e *CARP* sobre o sistema operacional *FreeBSD*. Esta solução tem como finalidade garantir que os serviços informatizados da instituição estejam disponíveis o máximo de tempo possível.

Capítulo 1

Introdução

A globalização e a consequente concorrência no mundo moderno dos negócios exige da área de tecnologia soluções mais seguras, tornando organizações e empresas cada vez mais dependentes destes recursos computacionais. Transações *on-line*, acessos a sistemas industriais e comerciais necessitam confiabilidade e disponibilidade cada vez maior.

Visando soluções viáveis para atender as necessidades das instituições em manter seus serviços virtuais disponíveis o máximo de tempo possível, surgiu a tecnologia de *Cluster*[1] de Alta Disponibilidade (*High Avaliability*). Esta tecnologia utiliza técnicas de redundância de *hardware*, garantindo que os serviços estarão disponíveis constantemente, pois a rede de dados e os sistemas nela disponíveis não dependerão exclusivamente de um único *hardware*.

Inicialmente, o Capítulo 2 apresenta a fundamentação teórica do sistema operacional *FreeBSD*, do protocolo de redundância de endereço *IP - CARP* e da *interface PFSync*, bem como seus funcionamentos e aplicações.

O Capítulo 3 descreve o ambiente que foi implementada esta pesquisa e método utilizado para sua implantação.

---

[1] Um *cluster*, ou aglomerado de computadores, é formado por um conjunto de computadores, que utiliza-se de um tipo especial de sistema operacional classificado como sistema distribuído. É construído muitas vezes a partir de computadores convencionais (*desktops*), sendo que estes vários computadores são ligados em rede e comunicam-se através do sistema de forma que trabalham como se fosse uma única máquina de grande porte.

Os resultados da implantação do *Cluster* de Alta Disponibilidade estão apresentados no capítulo 4.

1.1 Objetivo

Esta monografia tem como objetivo propor uma implementação para um ambiente real técnicas de *Cluster* de Alta Disponibilidade para *Firewall*, utilizando *PFSync* e *CARP* sobre o sistema operacional *FreeBSD*. Esta solução tem como principal finalidade de manter os serviços virtuais disponibilizados pela instituição o máximo de tempo possível, onde para isso, utilizam-se técnicas de replicação de arquivos e serviços, e redundância de *hardware*.

1.2 Motivação

Nesta monografia serão apresentadas técnicas de montagem e a implementação de um *Cluster* de Alta Disponibilidade na Universidade Vale do Rio Doce - UNIVALE demonstrando o aumento da confiabilidade e disponibilidade dos serviços prestados a toda comunidade acadêmica desta instituição.

Capítulo 2

Fundamentação Teórica

O *FreeBSD*, desenvolvido pela Universidade de *Berkeley*, é um sistema operacional *open source*, denominado *Software* Livre[1], do tipo *Unix*, descendente do *4.4BSD-Lite*[2].

> *Software Livre se refere à liberdade dos usuários executarem, copiarem, distribuírem, estudarem, modificarem e aperfeiçoarem o software [STALLMAN; RICHARD, 1984].*

Este sistema possui uma licença própria – *BSD*[3], onde estabelece que os créditos dos autores originais devem ser mantidos, ao contrário da licença *GPL*[4] da *FSF*[5], que não apresenta limitações para o uso do código. Ao desenvolver uma versão comercial de um programa sob a licença *BSD*, o profissional não tem a obrigação de disponibilizar o código fonte.

O foco principal do *FreeBSD*, segundo [Brown, 2004], é a performance, facilidade no uso e estabilidade, particularmente da pilha *TCP/IP*,[RFC 1180].

---

1 *Software* Livre é qualquer programa de computador que pode ser usado, copiado, estudado, modificado e redistribuído sem restrições.
2 *4.4BSD Lite* é a última versão do Sistema Operacional *BSD* original.
3 http://www.freebsd.org/copyright/index.html
4 http://www.gnu.org/copyleft/gpl.html
5 http://www.fsf.org

Uma vez o sistema instalado, torna-se facilmente realizar o *download*, compilar e instalar pacotes adicionais através de um sistema de *Ports*.

> *Ports é um avançado sistema de instalação de aplicativos no sistema operacional FreeBSD. Na maioria dos casos a aplicação é automaticamente baixada da Internet, adaptada e configurada, se necessário, compilada, instalada e registrada na base de registro de pacotes [BROWN, 2004].*

A coleção de *Ports* baixa e checa a integridade dos arquivos, cria a aplicação e instala. Isto simplifica toda a instalação, que é um dos aspectos que mais consome tempo no uso de aplicações em diferentes sistemas.

Está disponível nas seguintes plataformas: *Intel x86, DEC Alpha, Sparc, PowerPC e PC-98.* Assim como para as arquiteturas baseadas em processadores de 64 *bits*.

Devido a performance da pilha *TCP/IP*, o *FreeBSD* é uma excelente opção para *Firewall.*

> *O foco principal do FreeBSD é a performance, particularmente da pilha TCP/IP, com um número de companhias, incluindo Yahoo, usando o mesmo como servidor.[BROWN, 2004].*

Firewall, segundo [Zwicky, Cooper e Chapman 2000], é o nome dado ao dispositivo que tem por função encaminhar e/ou filtrar o tráfego entre redes distintas, impedindo a transmissão de dados nocivos ou não autorizados de uma rede a outra. Dentro deste conceito incluem-se, geralmente, os filtros de pacotes e *proxy* de protocolos.

É utilizado para evitar que o tráfego não autorizado possa fluir de um domínio de rede para o outro. O *Firewall* não possui capacidade de analisar toda a extensão do protocolo, ficando geralmente restrito a camada de transporte (nível 4) do modelo referência *OSI/ISO*.

Firewall existe na forma de *software* e *hardware*, ou na combinação de ambos. A instalação depende do tamanho da rede, da complexidade das regras que autorizam o fluxo de entrada e saída de informações e do grau de segurança desejado. Entretanto, existem 2 tipos de filtragem de pacotes no nível da camada de rede:

- *Stateless*[1] ou estático: cada pacote é analisado de forma independente, sem nenhuma associação com possíveis pacotes já processados. Esta opção é a mais simples, porém, consome mais recursos dos dispositivos;

- *Stateful*[2] ou dinâmico: são filtragens mais refinadas e que oferecem um desempenho visivelmente melhor. Nesta filtragem cada pacote é analisado e associado (ou não) a uma conexão já existente. Este processo permite que os pacotes associados às conexões estabelecidas passem automaticamente, diminuindo o *overhead*[3] de análise e ação sobre cada pacote.

Dentro das opções de *Firewall* disponíveis sobre o *FreeBSD*, será utilizado nessa pesquisa o *Packet Filter*, devido a sua integração com o protocolo *CARP*.

---

[1] http://en.wikipedia.org/wiki/Stateless_firewall

[2] http://en.wikipedia.org/wiki/Stateful_firewall

[3] *Overhead* é considerado qualquer processo ou armazenamento em excesso seja de tempo de computação, de memória, de largura de banda de rede ou qualquer outro recurso que seja requerido para ser utilizado ou gasto para executar uma determinada tarefa, e com a consequência a piorar o desempenho.

Packet Filter[1], referenciado como *PF*, foi originalmente desenvolvido por Daniel Hartmeier. É um sistema implementado inicialmente pelo *OpenBSD* para realizar filtragem e traduções de rede (*NAT*) em *TCP/IP*. O *PF* também é capaz de realizar normalização e condicionamento do tráfego *TCP/IP*, controlar banda e priorização de pacotes.

Através da utilização de uma *interface* de rede chamada *PFSync*, *Packet Filter State Table Loggin Interface*, é possível sincronizar as conexões estabelecidas através do *PF* entre os *Firewalls*. Monitorado através do `tcpdump`[2], são observados alterações na tabela de estados em tempo real. A *interface PFSync* pode enviar estas mensagens de alterações de estado para a rede de modo que outros nós rodando *PF* possam unir as alterações em suas próprias tabelas de estado.

Para solucionar o problema de repetições dos endereços *IP* é utilizado o *CARP*, C*ommon Address Redundancy Protocol*, que permiti que múltiplos *hosts* no mesmo segmento de rede compartilhem um endereço *IP*. Este grupo de *hosts* é referido como um grupo de redundância. O mesmo é atribuído a um endereço *IP* que é compartilhado entre os membros do grupo. Dentro do grupo, um *host* é designado o *master* e o restante como *backups*. O *host master* responde a qualquer tráfego ou requisições *ARP* direcionadas para o *IP* compartilhado. Cada *host* pode pertencer a mais de um grupo de redundância por vez.

Usualmente, o *CARP* é utilizado para criar um grupo de *firewalls* redundantes. O *IP* Virtual, que é atribuído ao grupo, é configurado nas máquinas clientes como o *gateway*[3] padrão. Caso o *Firewall* sofra uma falha, ou seja desligado, o *IP* se moverá para um dos *Firewalls backup* e o serviço continuará sem ser afetado.

---

1 http://www.openbsd.org/faq/pf
2 http://www.tcpdump.org
3 *Gateway* é uma máquina intermediária destinada a interligar redes, separar domínios de colisão e traduzir protocolos.

2.1 Estado da Arte

Atualmente encontra-se como proposta de protocolos para soluções de Alta Disponibilidade em *firewall* a implementação do *VRRP* ou *CARP*. Ambos protocolos tem por objetivo garantir a redundância de *hosts*, fazendo com que vários *hosts* sejam responsáveis por um único endereço em uma rede.

O *VRRP, Virtual Router Redundancy Protocol*, protocolo padrão do *IETF*[1], *Internet Engineergin Task Force*, é um software proprietário patenteado pela CISCO[2]. Este é protegido por patente e direitos autoriais. Por esse motivo a comunidade *open source* criou o padrão livre – *CARP*, um protocolo *FOSS*[3], nativo do *OpenBSD*.

Quando utiliza-se a tecnologia *VRRP* cria-se um endereço virtual vinculado a uma *interface* de rede, assim os *hosts* pertencentes ao mesmo grupo compartilham somente um mesmo endereço *IP*, a desvantagem deste protocolo é a atualização de todo o *cache ARP* sempre que um *host* mudar de estado [RFC 826].

Ao contrário do *VRRP*, o *CARP* trabalha com uma *interface* virtual, nomeada `carpN`, onde N é o número da *interface*, vinculada a uma *interface* real. A *interface* virtual tem um endereço *MAC*[4] Virtual, e todos os *hosts* que estão compartilhando o mesmo endereço também, assim quando um *host* muda de estado, o *cache* não precisa ser atualizado, pois o novo *host* mestre irá assumir não somente o endereço de *IP* mas também o endereço *MAC*. A única exigência para a utilização do *CARP* é a necessidade de no mínimo 3 endereços lógicos na mesma rede.

---

1 http://www.ietf.org
2 http://www.cisco.com
3 *FOSS (Free and Open-Source Software)* Software Livre e Código aberto.
4 *MAC (Media Access Control)* é o endereço físico da interface de rede. É um endereço de 48 bits, representado em hexadecimal.

2.2 *CARP*

2.2.1 Introdução ao *CARP*

CARP, *Common Address Redundancy Protocol*, é o Protocolo de Redundância de Endereço Comum. Seu objetivo principal é permitir que múltiplos *hosts*[1] no mesmo segmento de rede compartilhem um endereço *IP*. O mesmo é uma alternativa livre e segura ao *VRRP*, *Virtual Router Redundancy Protocol*, e ao *HSRP*, *Hot Standby Router Protocol*.

2.2.2 Uso do *CARP*

CARP é utilizado para criação de redes que necessitam de grupos de *firewalls* redundantes. É atribuído a um grupo de *firewall* um *IP* virtual que é configurado nas estações de trabalho da rede local como *gateway* padrão. Caso o *firewall* que detém o *IP* virtual sofra uma falha, ou seja desligado, outro *firewall* pertencente ao agrupamento será promovido a *master*, e este passará a responder como *gateway* padrão. Todo este procedimento é totalmente transparente aos usuários conectados à rede no momento.

[1] *Hosts* é todo e qualquer equipamento conectado a uma rede e que possua um endereço IP.

2.2.3 Funcionamento do *CARP*

CARP permite que um grupo de *hosts* no mesmo segmento de rede compartilhe o mesmo endereço lógico. Este grupo de *hosts* é referido como um grupo de redundância. Para o grupo de redundância é atribuído um endereço *IP* que é compartilhado entre os membros do grupo. Dentro do grupo, um *host* é designado o "*master*" e o restante como sendo "*backup"*. O *host master* é o que atualmente mantém o *IP* compartilhado, é o responsável por responder a qualquer tráfego ou requisições *ARP* direcionadas para o *IP* do grupo. Cada *host* pode pertencer mais que um grupo de redundância por vez.

O *host master* no grupo envia regularmente anúncios à rede local para informar aos *hosts backup* que o mesmo está ativo. Se os *hosts backup* não receberem um anúncio do *master* por um período de tempo pré-determinado, então um dos *hosts backup* que tenha configurado os valores advbase e advskew baixos será promovido a *master* do agrupamento de *firewalls* (Figura 2.1).

```
srvfwmaster# tcpdump proto carp
tcpdump: verbose output suppressed, use -v for full protocol decode
listening on vr0, link-type EN10MB (Ethernet), capture size 96 bytes
09:20:32.490966 IP srvfwmaster.fpf.univale.br > VRRP.MCAST.NET: VRRPv2,
Advertisement, vrid 1, prio 1, authype none, intvl 1s, length 36
09:20:33.495950 IP srvfwmaster.fpf.univale.br > VRRP.MCAST.NET: VRRPv2,
Advertisement, vrid 1, prio 1, authype none, intvl 1s, length 36
09:20:34.500939 IP srvfwmaster.fpf.univale.br > VRRP.MCAST.NET: VRRPv2,
Advertisement, vrid 1, prio 1, authype none, intvl 1s, length 36
09:20:35.505915 IP srvfwmaster.fpf.univale.br > VRRP.MCAST.NET: VRRPv2,
Advertisement, vrid 1, prio 1, authype none, intvl 1s, length 36
^C
4 packets captured
17 packets received by filter
0 packets dropped by kernel
srvfwmaster#
```

Figura 2.1: Tráfego *CARP* capturado via `tcpdump`.

É possível múltiplos grupos *CARP* existirem no mesmo segmento de rede. Anúncios *CARP* contêm o *Virtual Host ID* que permite membros do grupo identificar que grupo de redundância pertence.

Para prevenir que um usuário malicioso no segmento de rede falsifique anúncios *CARP*, cada grupo pode ser configurado com uma senha. Cada pacote *CARP* enviado ao grupo é então protegido por um *HMAC SHA1*[1].

2.2.4 Configuração do *CARP*

[1] *HMAC SHA1* é um algoritmo de autenticação chaveada de mensagem codificada

Para a utilização do *CARP* deve-se adicionar o suporte a interface, para isto, foi adicionado e recompilado nos dois servidores *firewall* o seguinte parâmetro no arquivo de configuração do *kernel*[2]:

```
device        carp
```

Figura 2.2: Parâmetro acrescentado ao arquivo de configuração do *kernel* para habilitar o reconhecimento de interface *CARP*.

Cada agrupamento de *firewall* é representado por uma *interface* de rede virtual. Assim, o *CARP* é configurado usando `ifconfig`.

```
ifconfig carpN create

ifconfig carpN vhid vhid [pass] [carpdev] [advbase] [advskew] [state]
endereço-ip máscara
```

Figura 2.3: Parâmetro do comando `ifconfig` para criação e configuração de uma *interface CARP*.

`carpN:` o nome da *interface* virtual, onde *N* é um inteiro que representa o número da *interface* (ex. `carp1`);

`vhid:` o virtual *host id*. Este é um número único que é usado para identificar o agrupamento de outros nós na rede. Valores aceitos são de 1 a 255;

[2] *Kernel* é o núcleo de um sistema operacional. Ele representa a camada mais baixa de interface com o hardware.

`pass:` a senha de autenticação usada quando estiver conversando com outros *hosts CARP* neste agrupamento. Deve ser a mesma para todos os membros do grupo;

`carpdev:` parâmetro opcional, especifica a *interface* de rede física que pertence a este grupo de redundância. Por padrão, o *CARP* tentará determinar qual *interface* usar procurando pela *interface* física que está na mesma subrede da combinação endereço-ip e máscara dada à *interface CARP*;

`advbase:` parâmetro opcional, especifica com que frequência, em segundos, será anunciado que o *host* pertence a um agrupamento de redundância. O padrão é 1 segundo. Valores aceitos são de 1 a 255;

`advskew:` parâmetro opcional, especifica o momento de desviar o `advbase` quando estiver enviando anúncios *CARP*. Manipulando o parâmetro `advbase`, o *host CARP master* pode ser escolhido. Quanto mais alto o número, menor a preferência pelo *host*, quando estiver escolhendo um *master*. O padrão é 0. Valores aceitos são de 1 a 254;

estado: força uma *interface CARP* a um certo estado. Estados válidos são *init*, *backup* e *master*;

endereço-ip: este é o endereço lógico atribuído ao grupo de redundância. O mesmo não tem que estar na mesma subrede que o endereço *IP* na *interface* física. Este endereço, contudo, precisa ser o mesmo em todos os *hosts* no grupo;

máscara: a máscara de subrede do *IP* compartilhado.

Além disso, através do `sysctl`, ferramenta disponível no *FreeBSD*, pode-se alterar o comportamento do *CARP* através das seguintes variáveis:

`net.inet.carp.allow:` permite a entrada de pacotes *CARP* ou não. Padrão é 1 (sim);

`net.inet.carp.preempt:` permite que os *hosts* dentro de um agrupamento apresentem um melhor `advbase` e `advskew` para assumir o lugar do *master*. Adicionalmente, esta opção habilita *failing over* em todas as *interfaces* caso uma *interface* fique indisponível (*down*). Se uma *interface* física *CARP* por algum motivo ficar indisponível, o *CARP* trocará o `advskew` para 240 em todas as outras *interfaces CARP*. Esta opção é 0 (desabilitada) por padrão;

`net.inet.carp.log:` registra em um arquivo de *logs* pacotes *CARP*. O padrão é 0 (desabilitado);

`net.inet.carp.arpbalance:` carrega balanceamento de tráfego entre múltiplos *hosts* no grupo de redundância. O padrão é 0 (desabilitado).

2.3 *PFSync*

2.3.1 Introdução ao *PFSync*

A *interface* de rede *PFSync* expõe certas alterações feitas na tabela de estados *PF*. Através da ferramenta de monitoramento `tcpdump` pode-se observar mudanças na tabela de estado. A *interface* `pfsync` envia mensagens

de alterações de estado para a rede de modo que outros nós rodando *PF* possam atualizar as alterações em suas próprias tabelas de estado.

2.3.2 Funcionamento do *PFSync*

A configuração padrão do *PFSync* é não enviar ou receber atualizações da tabela de estado das conexões de rede.

```
srvfwmaster#
srvfwmaster# tcpdump proto pfsync
tcpdump: verbose output suppressed, use -v or -vv for full protocol
decode
listening on vr0, link-type EN10MB (Ethernet), capture size 96 bytes
10:44:28.274131 IP 10.0.0.2 > 224.0.0.240: pfsync 20
10:44:28.809245 IP 10.0.0.1 > 224.0.0.240: pfsync 20
10:44:33.273171 IP 10.0.0.2 > 224.0.0.240: pfsync 20
10:44:33.809175 IP 10.0.0.1 > 224.0.0.240: pfsync 20
^C
4 packets captured
210 packets received by filter
0 packets dropped by kernel
srvfwmaster#
```

Figura 2.4: Tráfego *PFsync* capturado via `tcpdump`.

Quando o *PFSync* está configurado para enviar e receber atualizações na rede, o comportamento padrão é enviar atualizações *multicast*[1] na rede local. Todas as atualizações são enviadas sem autenticação. A prática mais comum neste procedimento é:

[1] *Multicast* é a entrega de informação para múltiplos destinatários simultaneamente onde as mensagens só passam por um *link* uma única vez.

- Conectar através de um cabo *crossover*[2] os dois nós que estarão trocando atualizações de estado das conexões e usar a *interface* como `syncdev`;

- Usar a opção `ifconfig` *`syncpeer`*, assim atualizações são direcionadas com *unicast* [3] para o nó, configurando o `ipsec` entre os *hosts* para proteger o tráfego *PFSync*.

2.3.3 Configuração do *PFSync*

Para a utilização do *PFSync* deve-se adicionar o suporte a *interface*, para isto, foi adicionado nos dois servidores *firewall* o seguinte parâmetro no arquivo de configuração do *kernel*:

```
device        pf
device        pfsync
```

Figura 2.5: Parâmetro acrescentado ao arquivo de configuração do *kernel* para habilitar o reconhecimento de interface *PFSync.*

Visto que o `pfsync` é uma *interface* de rede virtual, é configurado usando `ifconfig`.

[2] *Crossover* consiste na interligação de 2 computadores pelas respectivas placas de rede sem ser necessário a utilização de um concentrador.
[3] *Unicast* é quando um endereçamento para um pacote é feito a um único destino.

```
ifconfig pfsyncN syncdev [syncpeer]
```

Figura 2.6: Sintaxe do comando `ifconfig` para criação de uma interface *PFSync*

`pfsyncN`: o nome da *interface* `pfsync`. A `pfsync0` não existe por padrão quando estiver utilizando o *kernel GENERIC*;

`syncdev:` o nome da *interface* física usada para enviar atualizações `pfsync`;

`syncpeer:` parâmetro opcional que específica o endereço lógico de um *host* para trocar atualizações `pfsync`. Por padrão atualizações `pfsync` são *multicast* na rede local. Esta opção cancela este comportamento e em vez disso envia atualizações *unicasts* para o `syncpeer` especificado.

2.3.4 Integração entre *CARP* e *PFSync*

Combinando as características do *CARP* e *PFSync*, um grupo de dois ou mais *firewall* podem ser usados para criar um *cluster* de *firewall* completamente redundante e com alta disponibilidade. Veja um exemplo na figura 2.7.

- *CARP*: trata o *failover* automático de um *firewall* para outro;

- *PFSync*: sincroniza a tabela de estados entre todos os *firewalls* do agrupamento. Caso aconteça um *failover*, o tráfego pode fluir ininterrupto através do novo *firewall master*.

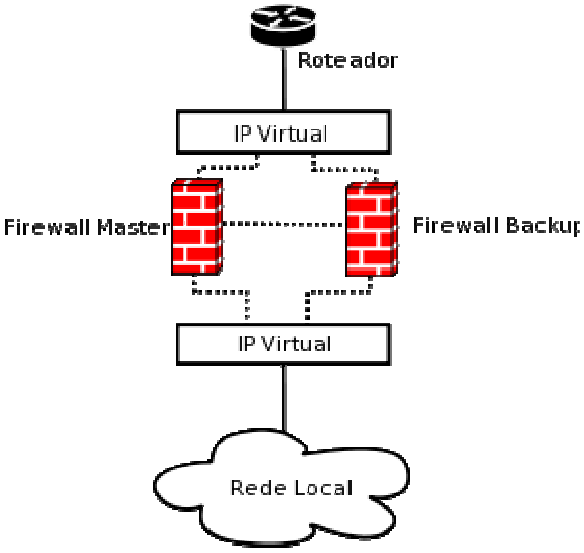

Figura 2.7: Uso do *CARP* com *PFSync*

Capítulo 3

Material e Método

3.1 Descrição do ambiente

Atualmente, a Universidade Vale do Rio Doce - UNIVALE possui aproximadamente 6.126 (seis mil e cento e vinte e seis) alunos e 1.188 (hum mil e cento e oitenta e oito) funcionários, entre corpo docente e técnico administrativo, totalizando 7.314 (sete mil e trezentos e quartoze) usuários dependentes da alta disponibilidade e confiabilidade de um sistema operacional.

Tabela 3.1: Tabela demonstrativa do número de usuários do sistema.

Distribuição	Total
Corpo Discente	6.126
Corpo Docente e Técnico Administrativo	1.188
Total Geral	**7.314**

Os dados da tabela 3.1 foram extraídos do banco de dados central da UNIVALE na data de 3 de fevereiro de 2006 e com aprovação do Gerente do Centro de Informática e Tecnologia – CIT desta instituição (Anexo 1).

No Centro de Informática e Tecnologia - CIT da Univale foi desenvolvido um sistema coorporativo onde todos os setores trabalham integrados, fornecendo e buscando informações em um banco de dados central.

Além da administração dos servidores de banco de dados, o CIT é responsável pelo acesso à internet de toda instituição e hospeda o Portal Univale. Nele estão disponíveis ferramentas que atendem a toda comunidade acadêmica.

O corpo discente tem acesso on-line a informações que competem a sua vida acadêmica e financeira dentro da instituição. O Portal do Aluno, disponibilizado no site da Univale, é uma ferramenta já consolidada que garante a agilidade nos processos exigidos diariamente para a formação do mesmo. São eles:

- Solicitação de documentos e declarações;
- Acesso ao corpo docente;
- Consulta e reserva de livros e outras referências bibliográficas;
- Consulta de notas e frequência;
- Situação financeira.

O Portal do Professor tem disponibilizado as seguintes ferramentas:

- Controle e lançamento de notas e frequência;
- Orientação em pesquisas;
- Lançamento do calendário de provas;
- Acesso ao corpo discente;
- Informativo a toda comunidade acadêmica.

Para o bom funcionamento dos sistemas desenvolvidos pelo CIT a toda comunidade acadêmica se torna um pré-requisito a alta disponibilidade e confiabilidade da rede de dados.

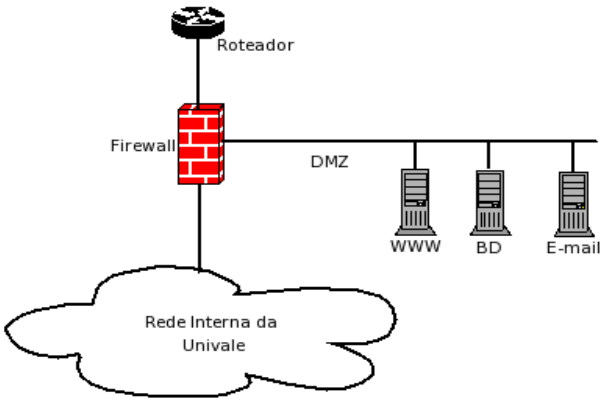

Figura 3.1: Diagrama de rede antes da implantação do *Cluster* de Alta Disponibilidade.

3.2 Descrição da Implementação

A implementação do *Cluster* iniciou-se pela preparação do *hardware* dos dois servidores que atuam como *firewall* da rede. Foram acrescentadas 4 (quatro) placas de redes - NIC, *Network Interface Card, fast-ethernet* em cada servidor com o objetivo de atender a estrutura de *IP* Virtuais exigidos pelo cenário apresentado (Figura 3.2).

O sistema operacional *FreeBSD* versão 6.1 foi instalado, os *firewalls* conectados diretamente através de um cabo *crossover* para sincronizar os estados de conexão entre os servidores.

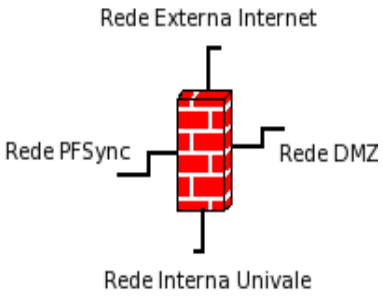

Figura 3.2: Conexões de redes do servidor *firewall*.

Com o objetivo de assegurar os estados das conexões da rede interna foi implantado a solução *PFSync.*

Foram definidos os endereços *IPs* virtuais e suas respectivas máscaras da seguinte forma:

- rede externa: 200.195.49.228 / 255.255.255.224;
- rede *DMZ*: 192.168.1.10 / 255.255.255.0;
- rede interna: 192.168.4.10 / 255.255.254.0.

Os *firewalls* foram denominados de *srvfwmaster* e *srvfwslave*, seus endereços lógicos foram definidos conforme segue:

- Endereços lógicos do servidor srvfwmaster:
    rede externa / *interface* rl0: 200.195.49.226 / 255.255.224.0
    rede *DMZ* / *interface* ed0: 192.168.1.254 / 255.255.255.0
    rede *PFSync* / *interface* rl1: 10.0.0.1 / 255.255.255.252
    rede interna / *interface* rl2: 192.168.4.1 / 255.255.254.0

- Endereços lógicos do servidor srvfwslave:

rede externa / *interface* rl0: 200.195.49.227 / 255.255.224.0
rede DMZ / *interface* rl1: 192.168.1.253 / 255.255.255.0
rede PFSync / *interface* rl2: 10.0.0.2 / 255.255.255.252
rede interna / *interface* rl3: 192.168.4.2 / 255.255.254.0

3.3 Configuração do *CARP* e *PFSync* nos servidores

3.3.1 Configuração do srvfwmaster

Habilitar a opção *preemption* e a *interface* para *failover* através do comando `sysctl`. Esta defini os computadores que fazem parte do agrupamento de *firewall*, o que estiver com o menor valor configurado no parâmetro `advskew` terá prioridade em assumir como *master* o grupo de redundância.

```
Srvfwmaster#
srvfwmaster# sysctl -w net.inet.carp.preempt=1
net.inet.carp.preempt: 0 -> 1
srvfwmaster#
```

Figura 3.3: Comando para habilitar o modo *preemption*

Configuração do *PFSync*:

```
srvfwmaster#
srvfwmaster#ifconfig pfsync0 syncdev vr0
srvfwmaster#ifconfig pfsync0 up
srvfwmaster#
```

Figura 3.4: Comando para definir e iniciar a *interface* `pfsync0` do *firewall master.*

Configuração *CARP* da rede externa:

```
srvfwmaster#
srvfwmaster#ifconfig carp0 create
srvfwmaster#ifconfig carp0 vhid 1 advskew 1 pass lash 200.195.49.228
255.255.255.224
srvfwmaster#ifconfig carp0 up
srvfwmaster#
```

Figura 3.5: Definição e configuração da *interface* `carp0`, rede externa do *firewall master*.

Configuração *CARP* da rede *DMZ*

```
srvfwmaster#
srvfwmaster#ifconfig carp1 create
srvfwmaster#ifconfig carp1 vhid 1 advskew 1 pass lash 192.168.1.10
255.255.255.0
srvfwmaster#ifconfig carp1 up
srvfwmaster#
```

Figura 3.6: Definição e configuração da *interface* `carp1`, rede *DMZ* do *firewall master*.

Configuração CARP da rede interna

```
srvfwmaster#
srvfwmaster#ifconfig carp2 create
srvfwmaster#ifconfig carp2 vhid 1 advskew 1 pass lash 192.168.4.10
255.255.254.0
srvfwmaster#ifconfig carp2 up
srvfwmaster#
```

Figura 3.7: Definição e configuração da *interface* `carp2`, rede interna do *firewall master*.

3.3.2 Configuração do servidor srvfwslave

Habilitar *preemption* e a *interface* para *failover* através do comando `sysctl`. A opção *preemption* definir que os computadores que fazem parte do agrupamento de *firewall*, o que estiver com o menor valor configurado no parâmetro `advskew` terá prioridade em assumir como *master* o grupo de redundância.

```
srvfwslave#
srvfwslave# sysctl -w net.inet.carp.preempt=1
net.inet.carp.preempt: 0 -> 1
srvfwslave#
```

Figura 3.8: Comando para habilitar o modo *preemption*

Configuração do *PFSync*:

```
srvfwslave#
srvfwslave#ifconfig pfsync0 syncdev vr0
srvfwslave#ifconfig pfsync0 up
srvfwslave#
```

Figura 3.9: Comando para definir e iniciar a *interface* `pfsync0` do *firewall slave.*

Configuração *CARP* da rede externa:

```
srvfwslave#
srvfwslave#ifconfig carp0 create
srvfwslave#ifconfig carp0 vhid 1 advskew 100 pass lash 200.195.49.228
255.255.255.224
srvfwslave#ifconfig carp0 up
srvfwslave#
```

Figura 3.10: Definição e configuração da *interface* `carp0`, rede externa do *firewall slave.*

Configuração *CARP* da rede DMZ:

```
srvfwslave#
srvfwslave#ifconfig carp1 create
srvfwslave#ifconfig carp1 vhid 1 advskew 100 pass lash 192.168.1.10
255.255.255.0
srvfwslave#ifconfig carp1 up
srvfwslave#
```

Figura 3.11: Definição e configuração da *interface* `carp1`, rede *DMZ* do *firewall slave.*

Configuração *CARP* da rede interna:

```
srvfwslave#
srvfwslave#ifconfig carp2 create
srvfwslave#ifconfig carp2 vhid 1 advskew 100 pass lash 192.168.4.10
255.255.254.0
srvfwslave#ifconfig carp2 up
srvfwslave#
```

Figura 3.12: Definição e configuração da *interface* `carp2`, rede interna do *firewall slave.*

Capítulo 4

Resultados obtidos

Toda infra-estrutura de rede exigida para o funcionamento dos serviços virtuais são diretamente dependentes do sistema de *firewall*.

Anteriormente à implementação do *CARP* e *PFSync*, caso ocorresse algum problema no *hardware* ou *software* deste servidor, todos os sistemas ficavam indisponibilizados – módulos acadêmico (Portal do Aluno e Professor), financeiro, administrativo.

Após a implementação do *Cluster* foi constatado o aumento da disponibilidade e confiabilidade dos serviços supracitados aos usuários da UNIVALE. Consequentemente não foi observado mais interrupções por motivos de falhas no sistema de *firewall*.

Foi observado também a significante economia de tempo e trabalho dispendidos ao Centro de Informática e Tecnologia após a implementação da solução proposta nessa pesquisa.

Capítulo 5

Conclusão

Após a implantação da solução de Alta Disponibilidade (*High Avaliability*) através da utilização simultânea dos protocolos *CARP* e *PFSync* houve o aumento da disponibilidade dos serviços oferecidos pela Universidade Vale do Rio Doce – UNIVALE, atendendo satisfatoriamente em tempo integral a todos os seus usuários.

Através desta pesquisa comprovou-se a facilidade de implementação desta tecnologia, que permite assegurar o funcionamento constante, através da redundância de *hardware* e *software*, do sistema de *firewall* - equipamento essencial para o funcionamento de uma rede de dados.

Outra vantagem desta solução é a facilidade de expansão do *cluster* em conjunto a rede de dados, bastando para isso o acréscimo de novos *hardwares* ao grupo de redundância. Esse processo é totalmente transparente aos usuários.

Como não há o alto investimento em licenças de *softwares*, optando pelo uso do *software* livre, o custo para a implantação deste *cluster* é consideravelmente baixo, visto seu retorno em segurança para a empresa.

É importante ressaltar que não há como quantificar o custo-benefício da implementação deste sistema para a empresa, já que este processo é de caráter preventivo.

Capítulo 6

Possibilidades para trabalhos futuros sobre o tema

Sendo soluções que oferecem segurança e disponibilidade, o *CARP* e *PFSyn* podem ser implementados em:

- redundância dos servidores de banco de dados da instituição pesquisada, objetivando a disponibilidade do acesso aos dados quando solicitados;
- balaceamento de carga entre servidores de *web* com grande volume de acesso.

E ainda como solução de *firewall*, a utilização do *PF* entre as redes administrativas e acadêmica da UNIVALE, com o objetivo de garantir a segurança de acesso à rede.

Referências Bibliográficas

BROWN, M. *Differentiating Among BSD Distros*. [on-line]. Disponível na Internet via www. url: http://www.serverwatch.com/tutorials/article.php/10825_3393051_2/. Arquivo capturado em 15 de março de 2006.

STALLMAN, R. What is Free Software?. [on-line]. Disponível na Internet via www. url: http://www.gnu.org/home.html. Arquivo capturado em 12 de março de 2006.

BROWN, M. *Differentiating Among BSD Distros*. [on-line]. Disponível na Internet via www. url: http://www.serverwatch.com/tutorials/article.php/10825_3393051_2/. Arquivo capturado em 15 de março de 2006.

WIKIPÉDIA, *Sistema de Ports*. [on-line]. Disponível na Internet via www. url: http://pt.wikipedia.org/wiki/Sistema_de_Ports. Arquivo capturado em 17 de março de 2006.

BORTOLIN, E. L. P. *Alta Disponibilidade usando CODA e LVS*. 2005. 85 f. Monografia (Pós-Graduação em Administração em Redes Linux) – Faculdade de Ciências da Computação, Universidade Federal de Lavras, Lavras.

OPENBSD, *OpenBSD Packet Filter*. [on-line]. Disponível na Internet via www. url: http://openbsd.default.co.yu/faq/pf/pt/carp.html. Arquivo capturado em 17 de março de 2006.

OPENBSD, *Redundância de Firewall com CARP e PFsync.* [on-line]. Disponível na Internet via www. url: http://openbsd.default.co.yu/faq/pf/pt/carp.html. Arquivo capturado em 17 de março de 2006.

RFC-826, An *Ethernet Address Resolution Protocol.* [on-line]. Disponível na Internet via ftp. url: ftp://ftp.rfc-editor.org/in-notes/rfc826.txt. Arquivo capturado em 17 de março de 2006.

RFC-1180, *A TCP/IP Tutorial.*. [on-line]. Disponível na Internet via ftp. url: ftp://ftp.rfc-editor.org/in-notes/rfc1180.txt. Arquivo capturado em 17 de março de 2006.

PITANGA, M. *Constuindo Supercomputadores com Linux*. 1. ed. São Paulo: BRASPORT, 2002.

PITANGA, M. *Computação em Cluster*. 1. ed. São Paulo: BRASPORT, 2004.

FILHO, N. A. P. *Linux, Clusters e Alta Disponibilidade.* 2002. Dissertação (Mestrado) – Faculdade de Ciências da Computação, Universidade de São Paulo, São Paulo.

PEREIRA, R. B. O. *Alta Disponibilidade em Sistemas GNU/LINUX*. 2005. 109 f. Monografia (Pós-Graduação em Administração em Redes Linux) – Faculdade de Ciências da Computação, Universidade Federal de Lavras, Lavras.

RIBEIRO, S. A. *Firewall em Linux.* 2004. 70 f. Monografia (Pós-Graduação em Administração em Redes Linux) – Faculdade de Ciências da Computação, Universidade Federal de Lavras, Lavras.

Anexos

Anexo A

Termo de Autorização

Declaro que conheço os requisitos da pesquisa "*ALTA DISPONIBILIDADE EM FIREWALL UTILIZANDO CARP E PFSYNC SOBRE FREEBSD*", desenvolvida na Universidade Vale do Rio Doce – UNIVALE, pelo pesquisador Marco Antônio Faria Botelho, e como esta instituição tem condições para o desenvolvimento da mesma, autorizo sua execução.

Governador Valadares, 28 de agosto de 2006.

_____

Euclides dos Santos
Gerente do Centro de Informática e Tecnologia