Uma implementação de alta disponibilidade em *Firewall* Linux

ALISSON MARQUES DA SILVA[1]
JOAQUIM QUINTEIRO UCHÔA[2]

[1]Fadom - Fac. Integradas do Oeste de Minas - CEP 35500-286 Divinópolis (MG)
alisson@fadom.br
[2]Curso ARL - DCC / UFLA - Cx Postal 3037 - CEP 37200-000 Lavras (MG)
joukim@ginux.ufla.br

Resumo: *A alta disponibilidade (High Availability – HA) visa aumentar a disponibilidade dos sistemas computacionais, através da redundância de recursos. Este artigo apresenta uma implementação de HA em um firewall Linux com Iptables e Squid, utilizando "computadores comuns" e software livre de maneira a manter um ambiente com alta disponibilidade e baixo custo.*

Palavras-Chave: *Alta disponibilidade, firewall, Linux, Heartbeat, mon.*

1 Introdução

Com a redução dos custos dos equipamentos e a crescente necessidade de armazenamento, processamento e recuperação das informações, os computadores têm se tornado cada dia mais importantes para as empresas, executando suas principais tarefas críticas. A interrupção dos serviços de um destes computadores pode causar grandes prejuízos, não somente financeiros, mas também danos à imagem e à confiança de clientes e investidores. De acordo com (RUIZ, 2000), em dois estudos patrocinados pela Oracle Corporation e pela Datamation, realizados nos Estados Unidos no ano de 1995, as empresas perderam, em média, de US$ 80 mil a US$ 350 mil por hora de paralisação não planejada. Estes estudos mostram a importância da disponibilidade dos sistemas computacionais para as empresas.

Diante desta crescente necessidade em manter o sistema disponível ininterruptamente, este artigo apresenta a implementação de um ambiente de alta disponibilidade em um *firewall*. O sistema será implementado utilizando-se computadores "comuns", o Linux como sistema operacional e *software* livre. Desta forma, pretende-se ter um *firewall* estável, seguro, com alta disponibilidade e baixo custo.

Para apresentação do trabalho, este texto encontra-se organizado como se segue: em um primeiro momento, na Seção 2, será apresentada uma introdução à disponibilidade de sistemas computacionais, destacando seus principais conceitos dando ênfase a alta disponibilidade. Posteriormente, será mostrada a implementação de alta disponibilidade em um *firewall* Linux, na Seção 3, com a instalação e configuração do sistema operacional e dos aplicativos. Depois disso, será apresentado um relato e análise dos testes e resultados

obtidos, na Seção 4. Por fim, na Seção 5 são apresentadas as principais conclusões a que foi possível chegar com o trabalho.

2 Disponibilidade de um sistema computacional

A disponibilidade de um sistema computacional é definida em (SZTOLTZ; TEIXEIRA; RIBEIRO, 2003), como sendo a probabilidade de um sistema estar funcionando e pronto para o uso em um dado momento. Esta pode ser calculada através da Equação 1, apresentada em (FILHO, 2004), em que A é a disponibilidade do sistema, MTBF é o tempo médio entre falhas (*Mean Time Between Failures*) e MTTR é o tempo máximo para reparos (*Maximum Time To Repair*):

$$A = \frac{MTBF}{(MTBF + MTTR)} \tag{1}$$

Segundo (RUIZ, 2000), um engano que geralmente ocorre no cálculo da disponibilidade é considerar somente as paradas não planejadas, ou seja, as que devem ser impedidas. Ele sugere que as paradas planejadas também sejam consideradas. Conclui-se que, quanto menor o número de paradas planejadas e não planejadas, melhor será a implementação. A Tabela 1, extraída de (FILHO, 2004), apresenta a relação entre o percentual de *uptime* de um sistema e o seu *downtime* anual e semanal.

Tabela 1: Tempo de Uptime x Downtime (FILHO, 2004)

Uptime	***Downtime*/ano**	***Downtime*/semana**
99	3,65 dias	1 hora, 41 minutos
99,9	8 horas. 45 minutos	10 minutos, 5 segundos
99,99	52,5 minutos	1 minuto
99,999	5,25 minutos	6 segundos
99,9999	31,5 segundos	0,6 segundos

A disponibilidade pode ser dividida em três classes (SZTOLTZ; TEIXEIRA; RIBEIRO, 2003), de acordo com a relação entre o tempo de *uptime*[1] e *downtime*[2] do sistema. São elas: disponibilidade básica, alta disponibilidade e disponibilidade contínua:

Disponibilidade Básica: é encontrada em computadores "comuns", sem a utilização de nenhum *hardware*, *software* ou dispositivo vinculado ao aumento de sua disponibilidade. A probabilidade deste sistema estar disponível para uso é de 99% a 99,9%.

Alta Disponibilidade: é alcançada através da utilização de recursos redundantes (*hardware*, *software*), visando eliminar as paradas planejadas e principalmente as não planejadas, e também os pontos únicos de falha (*Single Point Of Failure* – SPOF), que, de acordo com (RUIZ, 2000), são os recursos que não possuem alta disponibilidade. Deve ser lembrado que, uma falha em um destes SPOF, comprometerá a disponibilidade de todo o sistema. Estes devem ser eliminados, não somente nos computadores, mas em

[1] Tempo que o sistema está disponível para uso.
[2] Tempo que o sistema está indisponível.

toda infra-estrutura que lhe dá suporte, como rede, *switch*, roteador, *link* de Internet, energia elétrica, entre outros. A probabilidade de um sistema com alta disponibilidade estar disponível é, de no mínimo, 99,99%. Quanto maior esta probabilidade, melhor será a solução.

Disponibilidade contínua: na disponibilidade contínua, o grau de disponibilidade deve ser igual a 1, isto é, o sistema deve estar sempre *on-line*, sem nenhuma interrupção. Todas as paradas devem ser mascaradas, inclusive as paradas planejadas. Nenhuma parada pode ser perceptível para os usuários. Este tipo de disponibilidade é mais teórico; por mais que se aumente a disponibilidade, sempre haverá uma possibilidade de falha.

Existem alguns conceitos são importantes para o entendimento de alta disponibilidade e que, às vezes, não são utilizados de forma correta, causando confusão quanto ao seu emprego. Por isto, os principais conceitos sobre alta disponibilidade serão definidos a seguir:

Falha: acontece no universo físico, como uma oscilação de energia ou interferências eletromagnéticas.

Erro: uma falha pode acarretar um erro, que é a apresentação da falha no universo informacional. Uma falha pode gerar um erro em alguma informação. Por exemplo, um endereço de memória que seria 1 passa a ser 0.

Defeito: ocorre quando uma informação errônea não é percebida e tratada, se transformando em um defeito, como o travamento do sistema. O defeito é percebido no universo do usuário. Assim, uma falha no universo físico, pode causar um erro no universo informacional, que, por sua vez, pode causar um defeito no universo do usuário.

Failover: é quando uma máquina assume os serviços de outra. Geralmente ocorre após um defeito. O *failover* pode ser automático ou manual. Para uma solução de alta disponibilidade este deve ser automático.

Failback: é o processo de retorno de um determinado serviço para sua máquina de origem. A opção de executar ou não o *failback* vai depender da implementação, isto é, ele não é obrigatório.

Nodo: é cada computador que faz parte do *cluster* de alta disponibilidade.

Ter um sistema com alta disponibilidade não implica, necessariamente que ele seja tolerante à falhas. A tolerância à falhas visa impedir que as falhas se transformem em erros, e que esses se tornem defeitos. A alta disponibilidade, ao contrário da tolerância à falhas, não busca impedir que as falhas aconteçam, mas sim, mascarar as que venham a ocorrer, para que essas não sejam percebidas pelos usuários do sistema. Para isso, é necessário a redundância de recursos, pois, quando um destes apresenta uma falha, outro deve assumir suas atividades. A forma como a transferência da atividade de um recurso para outro acontece pode levar a grandes variações nos sistemas caracterizados como alta disponibilidade, de acordo com (FILHO, 2004). São elas:

Mascaramento Manual: é quando, na ocorrência de uma falha, é necessário a intervenção manual para colocar o componente redundante em funcionamento. Esse tipo de mascaramento não apresenta alta disponibilidade.

Cold Stand-By: neste tipo de transferência, ao ocorrer uma falha, os usuários são desconectados e perdem todo o trabalho que estava em progresso no momento da falha. O *failover* é automático, porém, o componente reserva estava desativado e precisa ser inicializado para começar a operar.

Warm Stand-By: assim como no *Cold Stand-By*, os usuários são desconectados e o *failover* é automático. No entanto, o componente *stand-by* já estava iniciado e em operação, possibilitando a recuperação parcial ou total dos trabalhos. O tempo de detecção da falha é o mesmo do item anterior, porém, o tempo de *takeover*[3] é reduzido drasticamente.

***Hot Stand-By* / Replicação Ativa:** os componentes são fortemente agrupados. Após uma falha, os usuários do componente defeituoso não são desconectados e não perdem os dados. Os usuários não percebem que o erro aconteceu, ou seja, a transferência é transparente, completa e a recuperação, instantânea.

3 Implementação

Após esta breve introdução à disponibilidade computacional, onde foram definidos seus principais conceitos, será apresentada a implementação de alta disponibilidade em um *firewall* Linux, conforme o ambiente representado pela Figura 1.

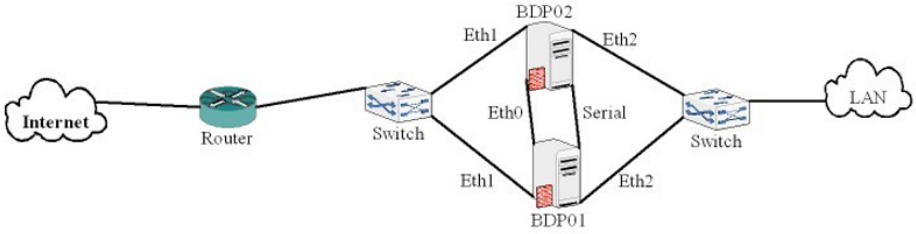

Figura 1: Estrutura da implementação de alta disponibilidade

Na implementação de alta disponibilidade em *firewall* apresentado na Figura 1, foram utilizados computadores "comuns", ou seja, sem nenhum *hardware* especial com o objetivo de aumentar sua disponibilidade. As máquinas são dois Pentium III, 750 MHZ, com 512 MB de memória RAM, disco rígido de 40 GB e 3 placas de rede Intel Pro 100 (uma para acesso a rede externa – Internet, outra para a rede interna e a terceira interligando as duas máquinas para o monitoramento entre nodos). Foram utilizados um cabo serial e um cabo *crossover*, para interligar os nodos.

O sistema operacional: foi utilizado nesta implementação o Debian[4] 3.1 com o *kernel* 2.4.27-2. O sistema de arquivos escolhido foi o ext3, devido ao seu sistema de *journaling*, que possibilita uma rápida recuperação em caso de falhas. As interfaces de rede foram configuradas conforme apresentado na Tabela 2. Observe que, por questões de segurança o endereço IP das interfaces de rede Internet foram ocultos. Além disso, nenhum destes

[3]Retorno a operação normal.

[4]Debian: `http://www.debian.org`.

endereços era o endereço do *cluster*, ou seja, o endereço pelo qual os usuários acessam a máquina. Este será configurado posteriormente.

Tabela 2: Configuração das interfaces de rede

Nodo	Nome	NIC	IP	Máscara	Rede
1	BDP01	eth0	10.10.10.1	255.255.255.252	Privada
		eth1	192.168.0.251	255.255.255.0	Local
		eth2	xxx.xxx.xxx.150	255.255.255.xxx	Internet
2	BDP02	eth0	10.10.10.2	255.25.255.252	Privada
		eth1	192.168.0.251	255.255.255.0	Local
		eth2	xxx.xxx.xxx.149	255.255.255.xxx	Internet

Para implementar esta solução foram utilizados quatro aplicativos: Iptables[5], Squid[6], Heartbeat[7] e mon[8], detalhados nas subseções a seguir. Nessa implementação não foi necessário nenhum aplicativo para sincronizar os dados entre os nodos. Isto, se deve ao fato dos arquivos de configuração não serem alterados com frequência. Caso seja necessário a sincronização dos dados pode ser utilizado o DRBD[9] ou Rsync[10].

3.1 Iptables

O Iptables é um *firewall* Linux, utilizado para prover segurança através da filtragem de pacotes. Também realiza o mascaramento de endereços IP, redirecionamento de portas e servidores. A versão utilizada neste trabalho foi a 1.2.11-10.

Para sua configuração foi criado um arquivo com o nome de *firewall* dentro do diretório `/etc`. Atribuiu-se permissão de leitura, escrita e execução somente para o `root`, através do comando `chmod 700 /etc/firewall`. Para que o *script firewall* pudesse ser executado durante a inicialização do sistema operacional, feito incluída sua chamada no arquivo `/etc/rc.local`. A Figura 2 apresenta um exemplo de *script firewall* que provê o acesso à Internet aos micros da rede interna, fazendo o mascaramento dos endereços IP e o redirecionamento do acesso à Internet para o *proxy* (Squid), localizados no próprio servidor.

3.2 Squid

O Squid é um *proxy* com *cache* utilizado para otimizar o acesso à Internet e controle de conteúdo. A versão do Squid utilizada neste trabalho foi a 2.5.9-10. O arquivo de configuração é o `squid.conf`, localizado no diretório `/etc/squid/`. Um exemplo deste arquivo contendo as partes que devem ser alteradas para seu funcionamento básico como *proxy* transparente pode ser visto na Figura 3.

[5]Iptables: `http://www.netfilter.org`.
[6]Squid: `http://www.squid-cache.org`.
[7]Heartbeat: `http://www.linux-ha.org/`.
[8]mon: `http://www.kernel.org/software/mon/`.
[9]DRBD: `http://www.drbd.org`.
[10]Rsync: `http://sunsite.dk/info/guides/rsync/rsync-mirroring.html`.

Figura 2: Arquivo `firewall`

```
IPTABLES="/usr/sbin/iptables"
$IPTABLES -F # Limpa as regras tabela Filter
$IPTABLES -t nat -F # Limpa as regras tabela NAT
# Mascara os ips internos para conexão a Internet
$IPTABLES -A FORWARD -s 192.168.0.0/24 -p tcp -j ACCEPT
$IPTABLES -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# Redireciona o trafego http para o Squid ? Proxy transparente
$IPTABLES -t nat -A PREROUTING -i eth1 -p tcp --dport 80 -j \
                        REDIRECT --to-port 3128
```

Figura 3: Arquivo `squid.conf`

```
# INSERT YOUR OWN RULE(S) HERE TO ALLOW FROM YOUR CLIENTS
acl rede scr 192.168.0.0/24
http_access allow rede
# HTTPD-ACCELERATOR OPTIONS
httpd_accel_host virtual
httpd_accel_port 80
# TAG: httpd_accel_with_proxy on|off
httpd_accel_with_proxy on
# TAG: httpd_accel_uses_host_header on|off
httpd_accel_users_host_header on
```

Como o foco deste documento não é a configuração do Squid e do Iptables, estes aplicativos foram apresentados com a configuração mínima, visando somente o funcionamento do ambiente de alta disponibilidade. Para otimizar os dois aplicativos, recomenda-se visitar o site oficial dos projetos e consultar (MARCELO, 2005) e (NETO, 2004).

3.3 Heartbeat

O Heartbeat é o aplicativo responsável pelo monitoramento da disponibilidade dos nodos. Ele envia sinais (*heartbeat – ping*) ao nodo que está sendo monitorado, através de uma conexão serial ou ethernet. Quando um *heartbeat* falha, o serviço entende que a máquina está inoperante. Um *failover* dos serviços da máquina é iniciado e o outro nodo assume seus serviços. Nesta implementação, serão utilizadas as duas conexões, serial e ethernet, para o monitoramento entre os nodos. Desta forma, caso algum problema venha a ocorrer em uma delas, como por exemplo, defeito na interface de rede utilizada para a comunicação entre os nodos, a conexão serial garantirá que esta comunicação não seja interrompida. Neste trabalho, foi utilizada a versão 1.2.3-9 do Heartbeat.

Antes de configurar o Heartbeat, é recomendado testar as conexões usadas por ele. A conexão serial pode ser testada no nodo `bpd01` através do comando "`cat < /dev/ttyS0`", e no nodo `bpd02` usando o comando "`echo "Teste conexao serial" > /dev/ttyS0`".

O texto deve ser recebido no nodo `bdp02`, caso contrário a saída serial e o cabo devem ser verificados. A conexão ethernet pode ser testada através do comando `ping` (`ping ip_outro_nodo`). Se o nodo em que foi executado o *ping* não receber a resposta do comando, a configuração, a interface de rede e o cabo devem ser verificados. Uma vez que as conexões entre as máquinas estão funcionando, inicia-se a configuração do aplicativo. Os arquivos de configuração encontram-se no diretório `/etc/ha.d/`. O primeiro arquivo a ser configurado é o `ha.cf`. Um exemplo do arquivo utilizado no nodo `bdp01` é apresentado na Figura 4.

Figura 4: Arquivo `ha.cf`

```
debugfile  /var/log/ha-debug  # Log de mensagens de depuracao - debug
logfile  /var/log/ha-log      # Log de mensagens gerais
logfacility  local0           # Envia mensagens de log e gerais para syslog
keepalive  2                  # Tempo entre os heartbeat em segundos
deadtime  30                  # Tempo para declarar o nodo inoperantede
warntime  10                  # Tempo de espera antes de emitir o aviso de
                              # "late heartbeat" (heartbeat atrasado)
initdead  120                 # Tempo necessário para a rede se tornar
                              # operante durante a inicialização.
auto_failback  off            # "on" habilita failback e "off" desabilita
baud  19200                   # Velocidade comunicacao serial
serial  /dev/ttyS0            # Porta serial
udpport  694                  # Porta udp utilizada  para bcast e ucast
ucast eth0 10.10.10.2         # Envia mensagens para ip especifico unicast
node  bdp01.fadom.com.br      # Nodos do cluster ? nome obtido através do
node  bdp02.fadom.com.br      # comando uname -n
```

Outro arquivo de configuração do Heartbeat é o `haresource` (Figura 5). Neste arquivo, especifica-se os serviços dos quais o *cluster* é o proprietário e o endereço de rede real do servidor. Este arquivo deve ser igual nos dois nodos.

Figura 5: Arquivo `haresource`

```
#nome cluster ip_externo_do_cluster ip_interno_do_cluster serviço
cache.ufla.br 192.168.0.254 xxx.xxx.xxx.151 squid
```

O último arquivo do Heartbeat a ser configurado é o `authkeys`, onde é indicada a forma de autenticação. Esta pode ser: CRC, método utilizado em redes seguras e que requer pouco uso dos recursos da CPU; MD5, requer mais recursos de CPU e oferece maior proteção do que o CRC; e SHA1, que consome mais recursos de CPU que os outros dois métodos, em contrapartida é o mais seguro. Será utilizado nesta implementação o CRC, pois a comunicação entre os dois nodos é segura. O arquivo configurado pode ser visto na Figura 6.

Figura 6: Arquivo `authkeys`

```
auth 1
1 crc
```

3.4 mon

O mon é um aplicativo para monitoramento de serviços, podendo ser configurado para envio de alertas (*e-mail*, SMS, entre outros), e para iniciar ou interromper outros aplicativos. Como o Heartbeat funciona trocando sinais entre os nodos, podem ocorrer situações onde um serviço importante não está funcionando e o *heartbeat* entre as duas máquinas está operante. O mon é utilizado para este tipo de situação, podendo finalizar o Heartbeat e provocar um *failover*, quando detectar o mal funcionamento de um determinado serviço. Com isto, o nodo primário é desativado e o nodo secundário assume seus serviços.

Seus arquivos de configuração encontram-se no diretório `/etc/mon/`, sendo que na maior parte dos casos o único arquivo que deve ser configurado é o `mon.cf`. Nesta implementação, o aplicativo, versão 0.99.2-8, está configurado para executar *ping* em dois grupos de servidores pré-definidos, como apresentado na Figura 7. Caso todos os servidores de um destes grupos pare de responder, o nodo é dado como isolado. Nesta situação, o mon envia um *e-mail* para os administradores do sistema e grava um *log* do evento. O computador começa a emitir um sinal sonoro e o Heartbeat é interrompido[11]. Como o Heartbeat pára de responder, o outro nodo dá este nodo como inoperante e assume seus serviços.

4 Testes Realizados

Para a realização dos testes foram simuladas algumas falhas nos recursos do ambiente. O objetivo foi analisar o comportamento do sistema quando um recurso falha. O tempo entre a ocorrência da falha, sua detecção e o restabelecimento dos serviços não foram considerados, pois estes podem ser ajustados nos arquivos de configuração. Para corroborar o resultado, todos os testes foram realizados no mínimo três vezes[12]. Os testes e os resultados obtidos são apresentados a seguir.

Queda no Nodo Primário: esse teste foi realizado desligando o nodo primário (cabo de energia desconectado), e, após 10 minutos o nodo foi reativado. Este teste simula a queda de energia, queima da fonte ou outro problema que desative o nodo. Na Figura 8 são apresentadas as três fases do teste. Após a queda do nodo `bdp01`, o nodo `bdp02` assumiu automaticamente os serviços. Quando o `bpd01` retornou, ele ficou em modo *stand-by*[13], e os serviços continuaram a ser oferecidos pelo `bdp02`.

[11] Para interromper o Heartbeat foi utilizado o *script* `heartbeat.alert` disponível em `http://ha.underlinux.com.br/heartbeat.alert`. Para que o computador emitisse o sinal sonoro, foi acrescentado no final do *script* a linha "`system ("/etc/ha.d/resource.d/AudibleAlarm start")`".

[12] Em todos os testes realizados o nodo `bdp01` foi considerado como primário.

[13] Caso seja necessário que o nodo primário assuma os serviços após recuperar-se de uma falha, é necessário alterar a opção `auto_failback` para "`on`" no arquivo `/etc/ha.d/ha.cf` nos dois computadores.

Figura 7: Arquivo `mon.cf`

```
# /etc/mon/mon.cf
cfbasedir        =/etc/mon
alertdir        =/usr/lib/mon/alerd.d
mondir             = /usr/lib/mon/mon.d
statedir             = /var/state/mon
maxprocs             = 20
histlength           = 100
historicfile         = /var/log/mon/mon.log
randstart            = 60s
hostgroup internet xxx.xxx.xxx.xxx                # ip do roteador
hostgroup local 192.168.0.4  192.168.0.5          # ip de servidores
watch internet
        service ping
                interval 5m
                monitor fping.monitor -a
                        alert mail.alert admin@dominio.com.br
                        alert heartbeat.alert
                        alert file.alert /var/log/mon.log
                        alertevery 1h
watch local
        service ping
                interval 5m
                monitor fping.monitor -a
                        alert mail.alert admin@dominio.com.br
                        alert heartbeat.alert
                        alert file.alert /var/log/mon.log
                        alertevery 1h
```

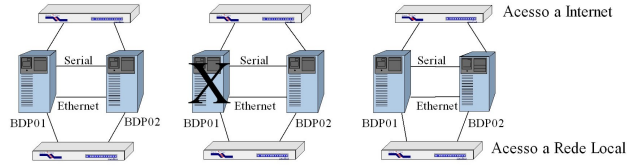

Figura 8: Queda no nodo primário

Falha em uma das interfaces utilizada pelo Heartbeat: essa falha foi por meio da desconexão de um dos cabos de comunicação entre os servidores (serial ou ethernet). Como pode ser visto na Figura 9, primeiro foi desconectado o cabo serial, depois esta conexão foi restabelecida. O mesmo processo foi repetido com a conexão ethernet. Esse ambiente foi implementado para suportar a falha em uma das interfaces de comunicação entre os nodos. Quando uma das interfaces falha, a comunicação (sinais do Heartbeat) deve continuar pela outra interface, o que ocorreu nos testes. Após uma

das interfaces falhar (serial ou ethernet), o sistema ficou estável, sendo oferecido no mesmo nodo que estava antes da falha, e, os sinais de *heartbeat* continuaram a ser enviados pela interface de comunicação ativa.

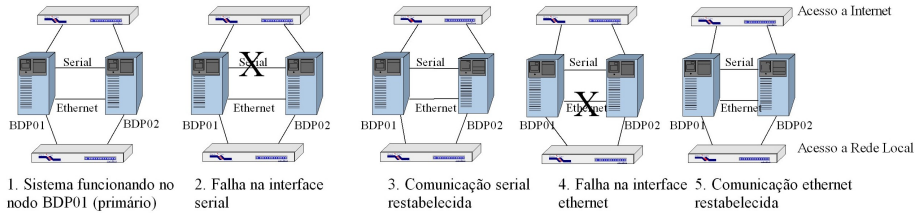

Figura 9: Falha em interface utilizada pelo Heartbeat

Nodo isolado: nesse ambiente, onde os nodos comunicam-se com duas redes distintas (rede local e Internet), considera-se que um nodo está isolado quando a comunicação com uma ou com as duas redes é interrompida. Esta falha foi simulada desconectando o cabo da placa de rede. Na Figura 10, são apresentados os passos que foram seguidos para a realização desse teste. Nas três formas testadas, o sistema respondeu conforme as expectativas. O Heartbeat do nodo foi desativado, um *log* do evento foi gerado e o computador começou a emitir um *bip* informando que o nodo está isolado. Com isso o nodo primário parou de responder aos sinais de *heartbeat*, e, o nodo secundário assumiu os serviços.

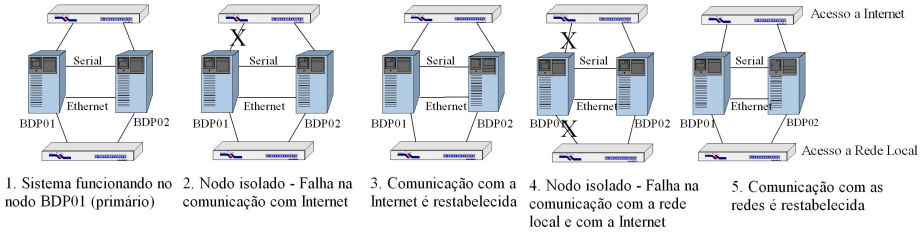

Figura 10: Nodo isolado

Falha nas duas interfaces utilizadas pelo Heartbeat: O ambiente foi implementado para restabelecer de falha em um dos recursos redundantes (solução com grau 1 de tolerância a falhas). Nesse teste foi analisado o comportamento do sistema quando as duas interfaces utilizadas pelo Heartbeat falham. A falha foi simulada desconectando os cabos, seguindo-se os passos representados na Figura 11. Quando as duas interfaces de comunicação falharam, o nodo `bpd02` (secundário) assumiu os serviços, mesmo com o nodo `bpd01`[14] não apresentando problemas. Quando a comunicação entre os nodos foi restabelecida o sistema ficou indisponível por alguns instantes, e, voltou a

[14] Este nodo fica com o endereço IP do *cluster* ativo, porém não responde a solicitações neste endereço. Desta forma, não são causados conflitos de endereço IP duplicados na rede.

ser oferecido pelo nodo `bpd01`. Também foi constatado nesse teste que, se nenhuma das conexões entre os nodos está ativa e o nodo `bdp02` (que assumiu os serviços após a perda da comunicação entre nodos) falhar, o sistema ficará indisponível até que o nodo `bdp02` retorne ou o `bdp01` seja reiniciado.

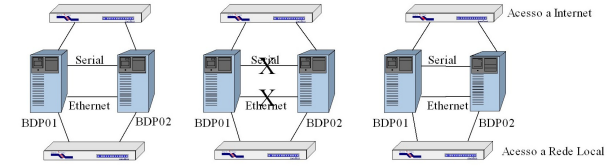

Figura 11: Falha nas duas interfaces utilizadas pelo Heartbeat

5 Conclusão

Alta Disponibilidade é uma área de grande interesse para o mercado corporativo. O sistema Linux apresenta um conjunto de ferramentas, que ainda estão em fase de amadurecimento mas que já apresentam resultados satisfatórios. As ferramentas utilizadas para a HA foram o Heartbeat e mon, configuradas para suportar a falha em um dos nodos da solução (solução com grau 1 de tolerância a falhas).

Os testes simularam falhas que podem ocorrer no dia a dia, como queima da fonte de alimentação, defeito nas interfaces de comunicação, entre outros. O sistema se comportou de maneira estável conseguindo se restabelecer das falhas de forma automática, sem a necessidade de intervenção do administrador. O sistema somente ficou indisponível quando ocorreram falhas simultâneas (falha nas duas interfaces utilizadas pelo Heartbeat), seguidas de uma falha do nodo que assumiu os serviços após a primeira falha. De uma forma geral o ambiente conseguiu atender ao que foi proposto.

Este trabalho mostrou a implementação de alta disponibilidade utilizando dois nodos, espera-se de trabalhos futuros a implementação deste tipo de solução com o uso de três ou mais nodos.

Referências

FILHO, N. A. P. *Serviços de Pertinência para Clusters de Alta Disponibilidade*. Dissertação (Mestrado) — Instituto de Matemática e Estatística da Universidade de São Paulo (IME/USP), 20 ago. 2004. Disponível em: <http://www.teses.usp.br/teses/disponiveis/45/45134/tde-04102004-104700/>.

MARCELO, A. *Squid: Configurando o Proxy para Linux*. São Paulo: Brasport, 2005.

NETO, U. *Dominando Linux Firewall Iptables*. São Paulo: Ciência Moderna, 2004.

RUIZ, A. Alta Disponibilidade em Servidores Linux. *Revista do Linux*, v. 6, fev. 2000. Disponível em: <http://labbi.uesc.br/apostilas/revista_do_linux/006/alta.html>.

SZTOLTZ, L.; TEIXEIRA, R. S.; RIBEIRO, E. de O. F. *Guia do Servidor Conectiva Linux 9*. Curitiba: Conectiva, 2003. Disponível em: <http://www.conectiva.com/doc/livros/online/9.0-/servidor/book.html>.