GABRIEL MADURO MARCONDES PEREIRA

SISTEMA DE E-MAILS DE ALTA DISPONIBILIDADE COM TRATAMENTO DE SPAM INTELIGENTE

INCONFIDENTES – MG
2013

GABRIEL MADURO MARCONDES PEREIRA

SISTEMA DE E-MAILS DE ALTA DISPONIBILIDADE COM TRATAMENTO DE SPAM INTELIGENTE

Trabalho de Conclusão de Curso apresentado como pré-requisito de conclusão do curso de Graduação em Tecnologia em Redes de Computadores no Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais – Câmpus Inconfidentes, para obtenção do título de Tecnólogo em Redes de Computadores.

Orientador: Thiago Caproni Tavares

INCONFIDENTES – MG
2013

GABRIEL MADURO MARCONDES PEREIRA

SISTEMA DE E-MAILS DE ALTA DISPONIBILIDADE COM TRATAMENTO DE SPAM INTELIGENTE

Data de aprovação: 07 de junho de 2013

Prof. Msc. Thiago Caproni Tavares
IFSULDEMINAS – Câmpus Inconfidentes

Prof. André Luigi Amaral Di Salvo
IFSULDEMINAS – Câmpus Inconfidentes

Prof. Luiz Carlos Branquinho Caixeta Ferreira
IFSULDEMINAS – Câmpus Inconfidentes

AGRADECIMENTOS

Agradeço primeiramente ao meu orientador, professor Thiago Caproni Tavares, que durante todo o período do curso contribuiu de forma imensurável em meus estudos e motivações, além de fornecer o suporte necessário para a conclusão deste trabalho. Ao colega de trabalho Rafael Gomes Tenório, pela contribuição na revisão de algumas partes do trabalho e pelas dicas de infinito valor. Agradeço ainda a todos os professores que contribuíram e contribuem para o sucesso do curso, muitas vezes prestando serviços além de sua obrigação. Da mesma forma, agradeço a toda instituição e envolvidos pela oferta e busca incessante pelo aprimoramento da qualidade de ensino na região.

RESUMO

O presente trabalho tem como objetivo estudar, implementar e testar um sistema de e-mails de alta disponibilidade, capaz de lidar de forma satisfatória com a ocorrência de *spam*, devendo este ser composto por implementações livres e padrões abertos. Os estudos levantados sob o tema indicam uma carência de documentação e experiências com esse tipo sistema. Dessa forma, o trabalho busca encontrar um conjunto de ferramentas que integradas sejam capazes de fornecer as características citadas. Para o requisito de alta disponibilidade, é utilizado um *cluster* de arquitetura GNU/Linux com armazenamento distribuído e sistema de arquivos compartilhado. O tratamento de *spam* é executado por um mecanismo probabilístico, capaz de aprender com seu uso e interação dos usuários. Com a conclusão dos trabalhos, verificou-se a viabilidade da implementação, além da eficácia do mecanismo de filtragem de *spam*, de forma que o sistema como um todo representa uma solução aceitável para o problema apresentado.

ABSTRACT

The present work aims to study, implement and test a high availability email system, able to satisfactorily deal with the occurrence of spam, and being composed by free and open standards implementations. The studies surveyed on the topic indicate a lack of documentation and experience with this type of system. Thus, the work seeks to find a set of integrated tools that are capable of providing the mentioned characteristics. For high availability requirement, a GNU/Linux architecture cluster is used with distributed storage and shared file system. Spam mitigation is executed by a probabilistic mechanism, able to learn from their use and user interaction. With the completion of the work, we verified the feasibility of the implementation, plus the effectiveness of spam filtering mechanism, so that the system as a whole represents an acceptable solution to the presented problem.

LISTA DE FIGURAS

LISTA DE ABREVIAÇÕES

- MTA – *Message Transfer Agent*
- MDA – *Message Delivery Agent*
- MUA – *Mail User Agent*
- IMAP – *Internet Message Access Protocol*
- POP3 – *Post Office Protocol version 3*
- SMTP – *Simple Mail Transfer Protocol*
- NTP – *Network Time Protocol*
- IP – *Internet Protocol*
- DRBD – *Distributed Replicated Block Device*
- HA – *High Availability*
- iSCSI – *Internet Small Computer System Interface*
- SAN – *Storage Area Network*
- OCFS2 – *Oracle Cluster File System version 2*
- GFS2 – *Global File System version 2*
- DLM – *Distributed Lock Manager*
- SPF – *Sender Policy Framework*
- DKIM – *DomainKeys Identified Mail*
- SAN – *Storage Area Network*
- NAS – *Network-attached Storage*
- DAS – *Direct-attached Storage*
- DNS – *Domain Name System*
- SA Forum – *Service Availability Forum*
- AIS – *Application Interface Specification*
- CRM – *Cluster Resource Manager*
- TLS – *Transport Layer Security*
- SSL – *Secure Sockets Layer*
- I/O – *Input/Output*
- RAID – *Redundant Array of Independent Disks*

SUMÁRIO

1. INTRODUÇÃO

Os sistemas de e-mails hoje são parte insubstituível de qualquer organização. Esse recurso possibilita a troca de informações de uma forma rápida, barata e eficiente, podendo substituir outros meios de comunicação. Os documentos transferidos através desse recurso podem ainda ter valor legal (GANDINI; SALOMÃO; JACOB, 2002), ocupando assim um papel ainda mais significativo.

Com tantas vantagens e possibilidades para os sistemas de e-mails, as instituições devem se perguntar: é necessário manter um sistema local? Essa pergunta é indispensável, principalmente nos dias atuais, quando contamos com uma vasta gama de soluções na nuvem, muitas delas gratuitas.

Algumas instituições não podem contar com essas soluções na nuvem, mesmo que pagas. Essa condição é consequência de algumas limitações de conformidade, impostas muitas vezes na tentativa de manter a segurança e soberania das informações.

Manter um sistema de e-mails próprio tem seu custo. Além da complexidade natural do sistema, é necessário pensar em outros fatores, entre eles segurança. A segurança da informação, muitas vezes resumida pelos princípios de confidencialidade, integridade e disponibilidade (PERRIN, 2008), deve envolver todo o sistema e informações nele contidas, além dos usuários, que direta ou indiretamente fazem parte do mesmo.

Um sistema de e-mails, composto por vários componentes, apresenta vantagens e desvantagens de segurança devido a sua composição. Por esses componentes serem separados, é possível tratar a segurança de cada um deles de forma isolada. Porém, quando se visualiza o sistema por inteiro, são vários componentes que se interligam, agregando complexidade à tarefa.

O fator disponibilidade do sistema é um requisito desse trabalho, de forma que toda a estrutura deve estar preparada para entregar os recursos e informações para os usuários, sempre que requisitado. Um serviço lento ou fora do ar causa insatisfação aos usuários, devendo assim ser evitado.

Apesar de alguns trabalhos tratarem de serviços e-mails de alta disponibilidade (CARTER; FINCH, 2004), a abordagem adotada neste é diferente. A ideia proposta consiste em colocar o serviço de e-mail sob uma plataforma conhecida e confiável de redundância, mantendo-a transparente para os componentes do sistema de e-mails. Além disso, as técnicas e mecanismos que são conhecidos e tem o bom funcionamento comprovado, são reutilizados.

A necessidade de alta disponibilidade em alguns tipos de serviços é alvo de pesquisas atuais. Alguns casos tratam de alta disponibilidade em serviços que não possuem esse recurso nativo (KÖVI; VARRÓ; NÉMETH, 2006), outros tratam da possibilidade aprimorar essa característica em serviços que oferecem mecanismos para redundância, mas que pode não atender a necessidade dos usuários (HAFERKAMP, 2011). As possibilidades são abundantes, sendo possível adicionar redundância em todos os níveis do sistema (HAAS, 2012b).

Mesmo com a infraestrutura segura e confiável, qualquer sistema de e-mails enfrenta um grande problema: a incidência de *spam*. Aquelas mensagens não solicitadas e que muitas vezes são enviadas para vários usuários representam uma ameaça para toda a organização. Esses e-mails transportam em si golpes, códigos maliciosos, propagandas, entre vários outros tipos de conteúdo que podem colocar em risco os ativos de uma instituição.

As técnicas de mitigação de *spam*, em sua essência, trabalham para que essas mensagens não solicitadas não cheguem aos usuários. Essas ferramentas podem trabalhar em níveis mais altos, outras em níveis mais baixos, na transmissão das mensagens ou diretamente em seu conteúdo, com ou sem a intervenção dos usuários finais.

Dessas técnicas, as mais avançadas são as baseadas em aprendizado através de inteligência artificial. Nesses casos, os sistemas identificam e aprendem os padrões dessas

mensagens, tornando-se capazes de identificá-las com índices de acuracidade diferenciados. Além disso, é possível que os usuários interajam com esses mecanismos, na tentativa de treiná-los para aprimorar seus filtros e consequentemente seus resultados.

O presente trabalho trata o assunto de sistemas de e-mail de alta disponibilidade com filtragem inteligente de *spam*. Por assim ser e a fim de garantir a integridade do conteúdo, o trabalho é organizado em três partes distintas: sistemas de e-mails, alta disponibilidade e filtragem inteligente de *spam*. Dessa maneira, se torna mais fácil e amigável apresentar os temas e seus detalhes mais importantes.

A fundamentação teórica, tratada no capítulo 2, discorre de forma resumida os princípios de cada um dos tópicos. Essas informações estão organizadas de forma sequencial e os tópicos mais significativos são expostos com mais detalhes e informações.

Por sua vez, a metodologia utilizada neste trabalho é discutida no capítulo 3. Nela são apresentados os objetivos do trabalho, a justificativa e relevância do tema. Além disso, o roteiro do trabalho é apresentado, juntamente com as técnicas e ferramentas utilizadas em sua execução.

A implementação da solução é tratada no capítulo 4. Esse capítulo apresenta a estrutura do sistema, a implementação do *cluster*, a implementação do sistema de e-mails e a implementação do mecanismo de filtragem de *spam*.

Por fim, são apresentadas as conclusões do trabalho e as contribuições obtidas durante o desenvolvimento da solução apresentada neste documento.

2. FUNDAMENTAÇÃO TEÓRICA

As informações apresentadas nesse capítulo constituem a fundamentação teórica acerca do tema do projeto. Alguns dos assuntos tratados nesse trabalho possuem ampla referência literária, outros não, como é o caso das soluções utilizadas. Apesar disso, os *sites* oficiais contam com documentação de qualidade que podem ser suficiente para a implantação dessas soluções.

A fim de garantir a organização dos assuntos e agrupar os tópicos relacionados, as informações foram organizadas em três itens: sistemas de e-mails, alta disponibilidade e filtragem de *spam*. As seções deste capítulo apresentam as informações sobre sistemas de e-mails em geral.

2.1. SISTEMAS DE E-MAIL

O termo servidor de e-mails pode se referir a uma máquina em particular num centro de processamento de dados que é responsável por enviar e receber os e-mails de seus usuários. Porém, um servidor de e-mails consiste numa variedade de componentes e programas, que usam vários padrões e protocolos para se comunicarem entre si (HEINLEIN, 2008).

Ao contrário da maioria das soluções de e-mails proprietárias, nas quais um único pacote de *software* lida com todas as operações, os sistemas de e-mails usados na Internet são compostos por vários padrões e protocolos que definem como uma mensagem é composta e como deve ser transferida de um remetente para um destinatário. Existem vários componentes de *softwares* envolvidos, cada um lidando com uma etapa na entrega das mensagens (DENT, 2003).

Sendo assim, neste trabalho, será adotado o termo sistema de e-mails para se referir a todos os componentes de *hardware*, *software*, padrões e protocolos que trabalham direta ou indiretamente para executar a tarefa de enviar e receber e-mails. Além disso, um sistema de e-mails pode ter o papel de armazenar por tempo indefinido as mensagens de seus usuários, o que adiciona uma camada extra de complexidade ao conjunto.

Os principais componentes e protocolos de um sistema de e-mails são apresentados na seção 2.1.1, acompanhados de uma breve descrição.

2.1.1. Componentes de um sistema de e-mails

O Agente de E-mail do Usuário ou MUA (acrónimo para *Mail User Agent*) é o *software* que interage com o usuário final, por onde ele envia e lê seus e-mails (DENT, 2003). Um MUA, também conhecido como cliente de e-mails, pode ser disponibilizado como um programa no computador ou como um *webmail*. Alguns MUAs locais contam com mecanismos de filtragem de *spam* que podem fornecer alguma flexibilidade para o usuário e até mesmo completar o mecanismo oferecido no servidor. Um *webmail* é um cliente de e-mail criado para funcionar através de um navegador (DENT, 2003). Essa alternativa é algumas vezes mais utilizada do que os clientes de e-mails comuns, instalados nos computadores dos usuários.

O Agente de Transporte de E-mail ou MTA (*Mail Transfer Agent*) é o componente responsável por enviar e receber os e-mails entre diferentes usuários e sistemas (DENT, 2003). Esse componente é o responsável pelo trânsito de mensagens pela Internet através do protocolo SMTP e usa padrões bem definidos de formatos de mensagens. O trabalho do MTA termina quando ele recebe um e-mail e entrega ao usuário, através de acesso direto a caixa de e-mails ou entregando a mensagem ao MDA.

O MDA ou Agente de Entrega de E-mail (*Mail Delivery Agent*) é o responsável por manter e prover acesso remoto às caixas de e-mails dos usuários (HEINLEIN, 2008). As caixas de e-mails podem utilizar vários formatos distintos para armazenamento e oferecer mais de uma maneira de acesso. Os protocolos POP3 e IMAP são utilizados para o acesso remoto as caixas postais nos servidores, cada um com suas particularidades. O IMAP é uma versão melhorada do POP3, fornecendo recursos avançados como sincronização das mensagens entre o servidor e o MUA (HEINLEIN, 2008).

Além dos componentes internos de um sistema de e-mail, a troca de mensagens pela Internet está intimamente relacionada com o DNS (*Domain Name System*). Quando um MTA tenta enviar um e-mail para outro domínio, ele faz uma consulta ao servidor com autoridade daquele domínio para descobrir o registro MX. Esse registro é composto pelo endereço do MTA que deverá receber os e-mails para aquele domínio e é através dele que se configura alguns mecanismos simples de redundância, como será detalhado no capítulo de implementação do sistema (KLENSIN, 2008).

2.1.2. Padrões e protocolos

Quando se fala em e-mails, o principal envolvido nessa tarefa é o SMTP. O objetivo do *Simple Mail Transfer Protocol* (SMTP) é transferir mensagens de forma eficaz e eficiente. Por padrão, o tráfego SMTP utiliza a porta 25/TCP (KLENSIN, 2008), mas esforços para reduzir a incidência de *spam*, alteraram o processo de submissão para utilizar a porta 587/TCP (ANTIspam.BR, 2013).

As mensagens trocadas entre os MTAs utilizando SMTP, são encaminhadas para os MDAs ou armazenadas diretamente nas caixas postais. Nos casos quando o MTA transfere a mensagem diretamente ao MDA, é comum o uso de uma versão simplificada do SMTP, o LMTP (MYERS, 1996).

Entre as opções para acesso às suas mensagens, os usuários podem utilizar o POP e IMAP. O *Post Office Protocol* ou POP é um protocolo que permite aos usuários acessarem e baixarem as mensagens de suas caixas postais (MYERS; ROSE, 1996). Dessa maneira, quando acessando a caixa postal, é possível baixar todas as mensagens e depois apagá-las do servidor, ou apenas baixá-las e manter uma cópia no servidor. Nesse caso, quando o usuário

necessitar acessar seu e-mail de um outro computador, suas mensagens estarão disponíveis.

Uma das desvantagens desse mecanismo é não ser possível manter caixas postais sincronizadas automaticamente em diferentes computadores. Quando, por exemplo, um usuário baixa uma mensagem em seu computador principal, essa mensagem não será apagada do servidor nem de seu notebook automaticamente e será baixada novamente quando o cliente de e-mails for sincronizado. Para resolver essa e várias outras deficiências, o *Internet Message Access Protocol* (IMAP) foi criado (CRISPIN, 2003).

O IMAP permite que os clientes de e-mails se sincronizem com o servidor, baixando apenas o cabeçalho das mensagens, por exemplo. A partir daí, quando as mensagens forem acessadas, o cliente baixa o conteúdo e apresenta aos usuários. Esse recurso é apenas um exemplo que apresenta o novo protocolo e seu alto grau de complexidade. Além de aprimorar vários recursos antes insuficientes no POP, o IMAP introduz novas funcionalidades que o tornam mais rico em recursos que seu antecessor.

Além desses protocolos, um *webmail* é muitas vezes disponibilizado para usuários. O *webmail* é um *software* escrito para funcionar num servidor, comunicando com o sistema de e-mails e apresentando aos usuários suas caixas postais e mensagens. Apesar de não ficar explícito para os usuários, os *webmails* utilizam os padrões anteriormente citados para acessar as caixas postais, operando de forma semelhante aos MUA locais.

Apesar de fora do escopo do trabalho, é importante citar que o *webmail* pode ser oferecido por uma infraestrutura diferente do sistema de e-mails, uma vez que o *software* utiliza os padrões conhecidos também oferecidos para os usuários. Essa organização é importante pois muitas vezes os *webmails* são oferecidos num pacote, em conjunto com agenda de contatos, calendários, comunicação instantânea, entre vários outros recursos que podem sobrecarregar a infraestrutura alocada para o sistema de e-mails.

Em todos os casos, é possível aprimorar a segurança nas trocas de informações entre usuários e sistema de e-mails através de protocolos criptográficos. O cenário mais comum é usar os protocolos de e-mail protegidos por um túnel fim a fim SSL/TLS. Dessa maneira, as informações são criptografadas na origem e descriptografadas no destino, tornando impossível interceptá-las. Outros mecanismos estão relacionados à autenticação e controle de acesso, que tem papel fundamental na segurança geral do sistema e das informações.

2.1.3. Software livre

No contexto de sistemas de e-mail, existem hoje diversas soluções livres para esse fim. A Figura 1 mostra o domínio das soluções de MTA livres num estudo de 2007 (SIMPSON; BEKMAN, 2007). O Sendmail, o MTA dominante segundo o estudo, ficou conhecido pelas diversas falhas de segurança, o que incentivou o desenvolvimento do Postfix.

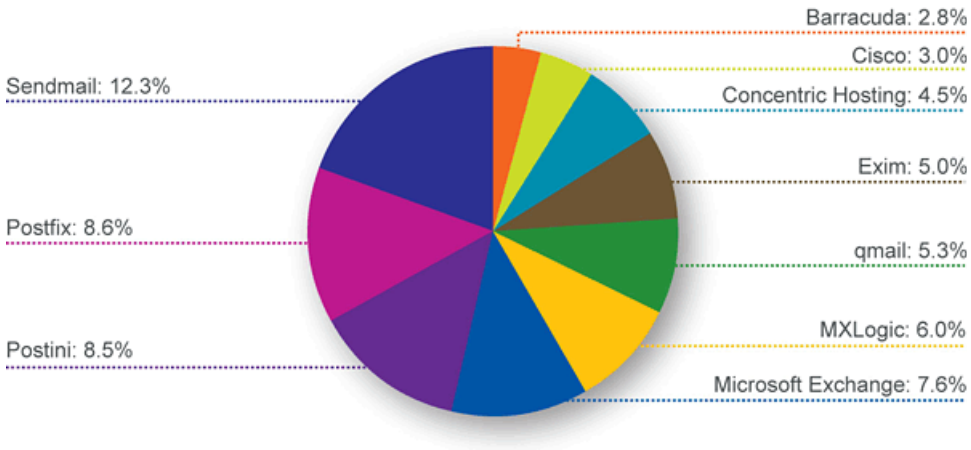

Figura 1: Os servidores de e-mails open source dominam o mercado.
(SIMPSON; BEKMAN, 2007)

Hoje, segundo estudo da *Security Space*, o cenário mudou. A Figura 2 mostra o histórico entre os anos de 2006 e 2012 sobre o uso de *softwares* MTA. É possível observar no gráfico que a solução Sendmail perdeu mercado para as alternativas mais novas e seguras.

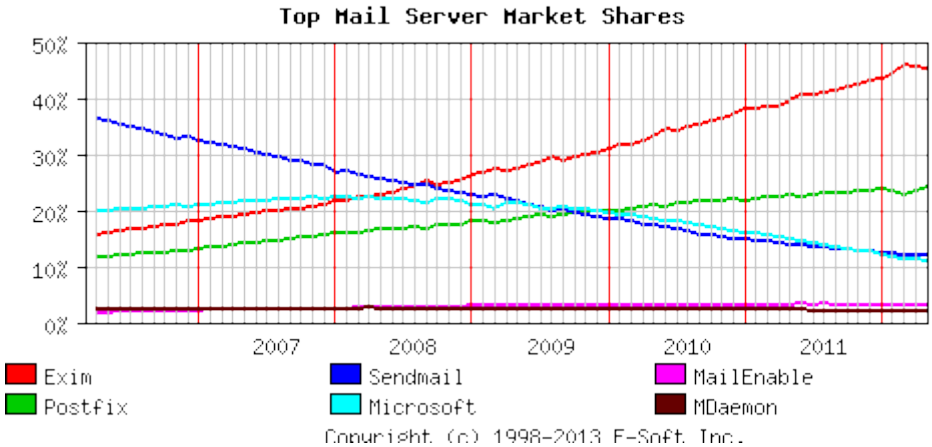

Figura 2: Os softwares servidores de e-mails mais utilizados no mercado.

(SECURITY SPACE, 2012)

É importante notar que, segundo as pesquisas, as soluções livres sempre dominaram o mercado de *softwares* servidores de e-mails. Além disso, várias soluções estão disponíveis, cada uma com suas particularidades, possibilitando ao administrador do sistema optar pela solução que melhor se adequa ao contexto e necessidade. No caso do *cluster*, como é utilizado um sistema de armazenamento distribuído, é importante que o MTA utilize o formato *Maildir* para armazenar os e-mails. Com esse mecanismo os e-mails são armazenados em arquivos individuais, deferentemente do formato *Mbox*, no qual as mensagens vão sendo adicionadas ao final de um único arquivo. Dessa forma, não é necessário utilizar bloqueio de acesso aos arquivos, como seria o caso se o formato *Mbox* fosse utilizado.

2.2. ALTA DISPONIBILIDADE

O conceito de alta disponibilidade refere-se à característica do sistema de se manter disponível por longos períodos de tempo sem interrupção. A disponibilidade, um dos princípios básicos da segurança, é a capacidade do recurso estar disponível quando o usuário precisa acessá-lo.

Essa característica pode ser alcançada de diversas maneiras, utilizando diversas soluções. Seguindo o princípio do trabalho de focar em soluções livres, as alternativas

baseadas na plataforma *GNU/Linux* serão tratadas em detalhes nesta sessão. Os conceitos relacionados são, em sua maioria, genéricos.

2.2.1. Cluster Linux

Um *cluster* consiste num conjunto de máquinas organizadas para trabalharem juntas, executando tarefas como se fossem apenas uma. Essas máquinas são chamadas de nós e pode ou não haver um controlador central, que dita as regras de operação dos integrantes do conjunto. Outra opção é que os próprios nós lidem com o gerenciamento, de forma que não haja pontos únicos de falha no sistema.

O projeto *Linux-HA* (LINUX-HA, 2013) fornece uma solução de alta disponibilidade para sistemas *Linux*, *FreeBSD*, *OpenBSD*, *Solaris* e *Mac OS X* fornecendo confiança, disponibilidade e operacionalidade. O principal produto do projeto é o *Heartbeat*, um programa gerenciador de *clusters*. O *Heartbeat* lida com as operações de comunicação e monitoramento entre os nós do *cluster*, além de fornecer alguns agentes que definem recursos a serem utilizados.

Outra solução para a camada de comunicação é o *Corosync* (COROSYNC, 2013). Esse *software* tem papel semelhante ao *Heartbeat*, porém é um projeto mais novo, conta com mais recursos e promete ser o futuro para a camada de comunicação em *clusters Linux*. O *Corosync* é derivado do projeto *OpenAIS* (OPENAIS, 2013), alteração a partir da qual alguns recursos passaram a fazer parte de cada um dos projetos separadamente (DAKE; CAULFIELD; BEEKHOF, 2008).

Além da comunicação entre os *hosts*, se faz necessário um componente que gerencie os recursos localmente, recebendo as informações da camada de comunicação e as traduza, manipulando os recursos e os processos. O *Pacemaker* (HAAS, 2012a) é um gerenciador de recursos de *cluster* (CRM – *Cluster Resource Manager*), capaz de alcançar alta disponibilidade para os serviços distribuídos, detectando e recuperando de falhas de *hardware* e de recursos, através de mensagens da camada de comunicação (*Heartbeat* ou *Corosync*).

A Figura 3 apresenta um modelo de *cluster*, composto por três nós com diferentes funcionalidades. O nó com a descrição "*www master*" é o fornecedor do serviço de

hospedagem de páginas e possui um endereço IP associado. O nó "*database master*" é o principal fornecedor do recurso banco de dados e também possui um endereço de rede. O nó "*standby machine*" funciona como um *backup*, que só entra em ação quando um dos nós principais falhar.

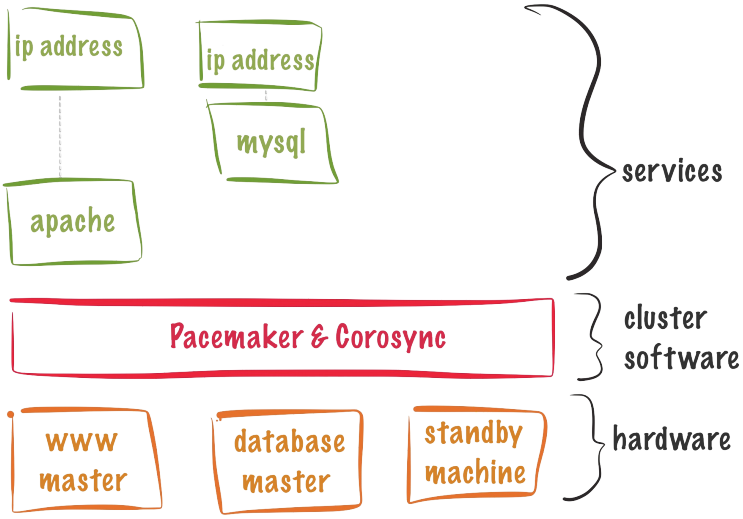

Figura 3: Arquitetura de um cluster com Corosync e Pacemaker.
(CLUSTER LABS, 2013)

Esse exemplo, apesar de simples, mostra a ideia por trás desse tipo de *cluster*. Além dos recursos do exemplo, vários outros agentes de recursos podem ser implementados, possibilitando ao *cluster* tratar cada recurso de acordo com as suas particularidades. Nos casos onde não haja agentes para um determinado recurso, ainda é possível fazer adequações para implantar o serviço (KÖVI; VARRÓ; NÉMETH, 2006).

A Figura 4 apresenta um dos outros possíveis modelos de *cluster* a ser implementado com as soluções citadas (CLUSTER LABS, 2012b). Nesse caso, o cenário é mais complexo, envolve mais recursos e mais nós.

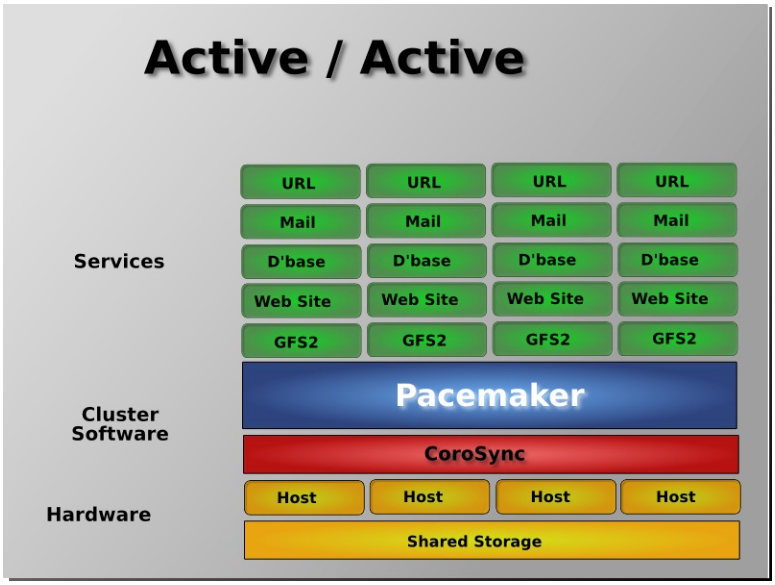

Figura 4: Cluster do tipo ativo/ativo.

(CLUSTER LABS, 2012a)

Nesse tipo de *cluster* todos os recursos ficam ativos em todos os nós simultaneamente, possibilitando não só a alta disponibilidade mais também o balanceamento de carga. O exemplo da Figura 4 ilustra a implementação de um *cluster* para hospedagem *web* com banco de dados e e-mail.

O modelo de *cluster* apresentado na Figura 4 levanta algumas preocupações sobre a integridade dos dados nos vários nós do modelo. Uma dessas preocupações se refere à situação na qual a conectividade entre os nós é interrompida. Nesse caso, se alguma providência não for tomada imediatamente, é possível que os nós passem a operar de forma ativa individualmente, comprometendo a integridade dos dados.

Para esses casos, existem mecanismos que detectam essa falha e através de configurações previamente definidas, isola determinados nós, garantindo que apenas um deles ou os que ainda estão interconectados, continuem operando e fornecendo os recursos. Após o reparo da conexão entre os nós, os dados são novamente sincronizados e o *cluster* retoma as operações normais (HAAS, 2012a).

A implementação de *cluster* pode ainda ser feita geograficamente, aproveitando essa característica quando disponível (BOOTH, 2013). Esse tipo de *cluster* tem variáveis diferentes das apresentadas aqui e se baseia num mecanismo de *tickets* para gerenciar os recursos.

2.2.2. RAID

A conhecida ferramenta RAID também é muito útil nesse caso. Durante esse trabalho foram utilizadas máquinas virtuais para a implementar e testar os componentes do sistema, porém, caso sejam usadas máquinas físicas, replicação de discos com o RAID 10 pode ser utilizada. O RAID 10, por exemplo, que espelha os discos e os integra de forma que as operações de escrita possam ser distribuídas entre os discos do conjunto, fornece segurança e desempenho simultaneamente. Apesar de haver outras alternativas de RAID, nesse caso, o RAID 10 parece ser a opção mais viável.

2.2.3. LVM

O *Logical Volume Manager* (LVM) fornece uma camada adicional de flexibilidade no armazenamento. Integrado ao *kernel* do *Linux*, o LVM permite que sejam criados e gerenciados volumes virtuais, representando partições. Assim, é possível gerenciar grandes volumes de armazenamento adicionando e removendo discos, ou simplesmente criar uma participação, o que sem o LVM seriam partições independentes.

No caso de um servidor de e-mails, no qual todos os usuários manterão de forma hierárquica todos seus dados numa mesma partição, pode se tornar um inconveniente gerenciar os discos e partições. Algumas soluções não possibilitam armazenar caixas postais em partições diferentes, o que nesse caso, se torna um impedimento. Com o LVM é possível criar uma única partição com vários discos. Ao considerar os benefícios do RAID com o LVM, tem-se uma solução de armazenamento flexível e segura.

Outro recurso importante do LVM é a capacidade de criar *snapshots*. Um *snapshot* pode ser comparado a uma foto, um registro de uma partição em um determinado momento.

Nos casos de *backups online*, aqueles nos quais o recurso não é interrompido para a cópia do dos dados em uso, o problema de corromper as informações é uma realidade.

Durante a cópia das caixas postais dos usuários para *backup*, por exemplo, é possível que após iniciada a cópia, alguma alteração seja feita nos dados, podendo comprometer a integridade de todo o conjunto. Com um *snapshot*, as novas alterações passam a ser feitas numa área temporária, sem afetar os dados originais e sua integridade. Após a conclusão do *backup*, as modificações são fundidas aos dados originais, tudo de forma transparente aos usuários e processos do sistema.

2.2.4. DRBD

O DRBD é um sistema de armazenamento distribuído para a plataforma GNU/Linux. De certa forma, o DRBD é semelhante ao RAID 1, porém a replicação dos dados é feita através da rede. Dessa forma, dois discos em diferentes máquinas numa rede estão sempre replicados (DRBD, 2013).

A Figura 5 apresenta os detalhes de funcionamento do DRBD e todos os componentes envolvidos.

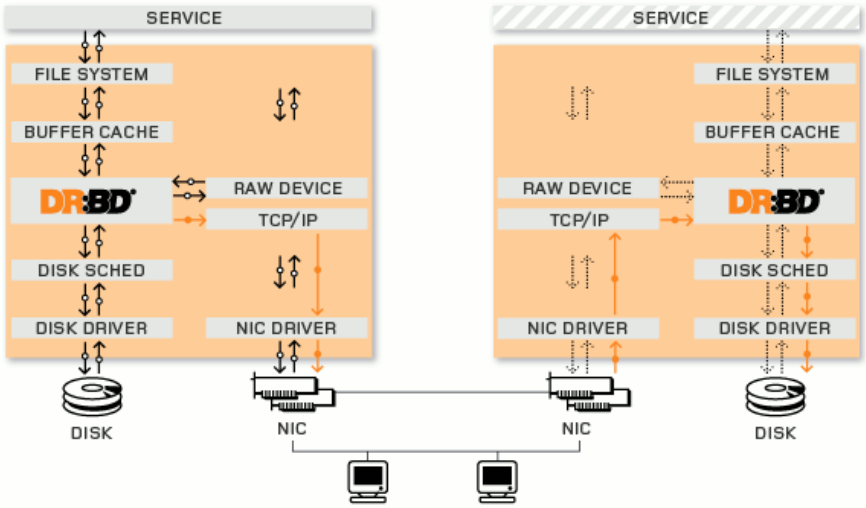

Figura 5: Diagrama de funcionamento do DRBD.

(DRBD, 2013)

O DRBD opera em uma camada baixa, sem lidar com o sistema de arquivos. Ao se utilizar um sistema de arquivos comuns (ext4, por exemplo) não é possível montar o volume em duas máquinas distintas simultaneamente. Para tal, é necessário o uso de um sistema de arquivos distribuído, como o OCFS2.

2.2.5. OCFS2

O OCFS2 (*Oracle Cluster File System*) é um sistema de arquivos de *cluster* com discos compartilhados, capaz de fornecer alta performance e alta disponibilidade. Como ele fornece uma semântica de sistema de arquivos local, pode ser utilizado com praticamente todas aplicações. As aplicações de *cluster* podem fazer uso das operações de *I/O* paralelas através de vários nós, para dimensionar facilmente as aplicações. Outras aplicações podem fazer uso das facilidades do sistema de arquivos para garantir o *failover* de aplicações no caso de algum dos nós falhar (PROJECT OCFS2, 2013).

Além de todos os recursos oferecidos pelo OCFS2 que garantem o bom funcionamento do sistema de arquivo num *cluster*, o DLM (*Distributed Lock Manager*) garante que as aplicações que necessitam de acesso único e exclusivo a determinado arquivo, possam funcionar sem problemas. Dessa forma, esses arquivos não são acessados simultaneamente em dois nós diferentes de um *cluster*, por exemplo, e a integridade do arquivo é garantida (FASHEH, 2006).

Nessa sessão foram apresentados de maneira resumida os conceitos e ferramentas que podem compor um *cluster* na plataforma GNU/Linux. Existem também outros projetos semelhantes, que não foram citados no trabalho por não fazerem parte da implementação, como detalhado no capítulo 4. A seção 2.3 apresenta as informações sobre filtragem de *spam*.

2.3. FILTRAGEM DE SPAM

Segundo o projeto Spamhaus (SPAMHAUS, 2013a), “o termo *spam* é utilizado para se referir ao envio em massa de e-mails não solicitados. Não solicitado significa que o destinatário não forneceu nenhuma permissão verificável para que a mensagem fosse enviada a ele. O envio em massa se refere às mensagens que são enviadas como parte de uma grande coleção, todas tendo conteúdo substancialmente idêntico”. Ao contrário do *spam*, o termo *ham* é utilizado para relacionar as mensagens legítimas, que não são consideradas *spam*.

Uma mensagem é *spam* apenas se ela não foi solicitada e foi enviada em massa. O envio de e-mail não solicitado é normal, como nos casos de primeiros contatos de venda, consultas de emprego, consulta de vendas, entre outros. O envio de e-mails em massa também é normal, como nos casos de assinaturas de *newsletters*, listas de discussão, comunicação com clientes, entre outros casos (SPAMHAUS, 2013a).

Segundo os dados do *Internet Security Threat Report 2013 - Volume 18* da *Symantec Corporation*, a estimativa global de envio de *spam* por dia é de 69% para o ano de 2012, o que corresponde a 30 bilhões de mensagens por dia. Nos anos anteriores, o número era maior. Um dos fatores que contribuem para essa queda é uso redes sociais para propagação de *malwares* e *phishing*, uma vez que essas redes são muito utilizadas pelos usuários (SYMANTEC, 2013).

2.3.1. Técnicas de mitigação de spam

A mitigação de *spam*, assunto alvo de constante pesquisa, é executada hoje por diversos mecanismos e técnicas. Algumas dessas são protocolos, que operam de forma integrada aos servidores e aos protocolos básicos da Internet. A seção 2.3.1.1 apresenta uma breve descrição das técnicas e mecanismos mais utilizados.

2.3.1.1. SPF

A atual infraestrutura de e-mail tem uma característica que qualquer *host* pode injetar e-mails num determinado sistema, identificando-se através de qualquer domínio. Os *hosts* podem fazer isso em vários níveis: em particular, na sessão, no envelope e no cabeçalho do e-mail. Apesar de esse recurso ser desejável em algumas circunstâncias, ele é o maior obstáculo na tentativa de reduzir e-mails não solicitados, ou *spam* (WONG; SCHLITT, 2006).

O SPF é um protocolo que permite que detentores de domínios autorizem determinados *hosts* a usarem seu nome de domínio nos campos "MAIL FROM" e/ou "HELO". Os detentores de domínios em conformidade com o protocolo publicam registros SPF especificando quais *hosts* são permitidos a utilizarem seus domínios, e os destinatários que também utilizam o SPF fazem uso dos registros publicados para testar a autorização dos MTA (WONG; SCHLITT, 2006).

2.3.1.2. DKIM

O padrão DKIM, ou *DomainKeys Identified Mail*, define um mecanismo pelo qual as mensagens podem ser assinadas criptograficamente, de forma que um MTA possa assegurar a responsabilidade por aquela mensagem. O funcionamento do mecanismo é baseado em criptografia assimétrica, disponibilizando a chave pública do domínio através de um registro no DNS (HANSEN; CROCKER; HALLAM-BAKER, 2009).

Dessa maneira, quando um MTA envia uma mensagem, ele adiciona ao seu cabeçalho um campo “DKIM-Signature” que possui as informações sobre a assinatura do e-mail. Quando o MTA do destinatário recebe a mensagem, ele pode consultar o servidor de nomes com autoridade sob aquele domínio, descobrir a chave pública e se certificar que aquela mensagem originou-se daquele sistema (HANSEN; CROCKER; HALLAM-BAKER, 2009).

2.3.1.3. Whitelisting

As técnicas de *whitelist*, ou lista branca, são aquelas nas quais se mantém uma base de dados com os *hosts* confiáveis. Assim, quando um e-mail é recebido, seu remetente é comparado aos domínios na *whitelist*. Se encontrado, pode significar ser um endereço confiável ou até que aquele e-mail não precisará de tratamento adicional.

Além das listas mantidas localmente, é possível baixar essas listas pela Internet. Algumas empresas disponibilizam e até comercializam *whitelists* compostas por domínios previamente examinados. Um exemplo conhecido é o Spamhaus Whitelist (SPAMHAUS, 2013b).

2.3.1.4. Blacklisting

A *blacklist* é o oposto de uma *whitelist*. Esse tipo de lista mantém os endereços não confiáveis. Essas bases de dados podem ser criadas previamente pelo administrador do sistema ou mesmo copiada da Internet. Existem serviços que disponibilizam essas listas através de assinaturas.

2.3.1.5. Greylisting

Ao contrário das *blacklists* e das *whitelists* que identificam como bons ou ruins os servidores de e-mails, a técnica *greylisting* tem uma abordagem diferente. Esse mecanismo provê um serviço (recebendo um e-mail, por exemplo) à um servidor não conhecido ou suspeito, degradando o serviço, de forma que esse servidor receba uma mensagem de falha temporária (KUCHERAWY; CROCKER, 2012).

Apesar de hoje os mecanismos de envio de *spam* estarem altamente sofisticados, ainda existem alguns que, ao receberem essa mensagem de recusa, não tentam enviar novamente. Ao contrário de outras soluções de filtragem de *spam*, que podem ser caras e

complexas, a *greyliting* é um mecanismo barato e é considerado essencial ao repertório das atuais técnicas de mitigação de *spam* (KUCHERAWY; CROCKER, 2012).

2.3.1.6. Filtragem por conteúdo

Apesar da disponibilidade de vários mecanismos de filtragem de *spam*, eles não são suficientes para lidar com o problema. O *spammers*, como são conhecido os indivíduos que enviam *spam*, criam novas soluções para contornar cada novo mecanismo de filtragem. O *spam* hoje faz parte de um complexo sistema, envolvendo criminosos com *botnets* e mecanismos extramente sofisticados e de difícil identificação.

Nos casos em que todos os mecanismos em operação não conseguem lidar com o *spam*, ainda há uma última alternativa: os filtros baseados em conteúdo. Esse tipo de filtro leva em consideração o conteúdo do cabeçalho e do corpo da mensagem.

As opções mais simples são aquelas que mantém regras estáticas pré-definidas. Assim, quando uma mensagem passa pelo filtro, ela é comparada à determinados padrões e se houver uma certa compatibilidade, a mensagem será tratada como *spam*. Esse tratamento pode variar desde a marcação de um campo no cabeçalho até a exclusão definitiva da mensagem.

Apesar de cumprir seu papel e filtrar as mensagens, esse tipo de filtro tem um problema grave. Os *spammers* examinam essas regras e criam suas mensagens de forma que não sejam classificadas como *spam* pelos filtros.

Além dos filtros com regras estáticas, existem outras soluções mais inteligentes. Esses filtros utilizam de técnicas de aprendizado de máquina para decidir se uma mensagem é ou não *spam*, muitas vezes, com resultados satisfatórios.

2.3.2. Filtragem inteligente

A principal diferença entre um mecanismo de filtragem que utiliza apenas regras estáticas pré-definidas e um mecanismo inteligente, é a capacidade de aprender com as

situações e informações pelas quais é submetido. O trabalho de (BLANZIERI; BRYL, 2008) apresenta uma pesquisa com mais informações sobre essas técnicas, incluindo os detalhes sobre cada uma delas.

Uma das técnicas de aprendizado utilizam o conceito de redes bayesianas, descrita no trabalho de (GRAHAM, 2003a). Através de técnicas probabilísticas, o mecanismo é capaz de identificar a probabilidade de uma mensagem ser ou não *spam*, de acordo com o seu conteúdo. Para tal, esse tipo de mecanismo precisa ser treinado a fim de aprender a classificar as mensagens.

Esse treinamento consiste em submeter várias mensagens conhecidas ao mecanismo de treinamento, informando se aquela mensagem é ou não *spam*. Dessa maneira, o mecanismo separa as palavras das mensagens, também chamados de *tokens*, adicionando-as a um banco de dados. Nesse banco de dados também consta a frequência com que essas palavras aparecem em cada tipo de mensagem. Com essas informações, o mecanismo é capaz atribuir uma probabilidade para as mensagens, no intuito de classificá-las como *spam* ou *ham* (GRAHAM, 2003[a]).

A partir desse princípio, várias ferramentas foram criadas, cada uma com modificações que podem apresentar resultados diferenciados. Um exemplo, é como são tratados os *tokens* que podem ser compostos por apenas palavras, *tags* HTML ou campos do cabeçalho da mensagem. Além desses, vários outros atributos podem ser optimizados a fim de se obter melhores resultados (GRAHAM, 2003b).

Concluída a fundamentação teórica dos assuntos relacionados, o capítulo 3 apresenta os detalhes sobre a metodologia adotada durante a execução do trabalho.

3. METODOLOGIA

Após os estudos preliminares, os esforços envolveram encontrar e avaliar as alternativas disponíveis para atender os objetivos do trabalho. O maior desafio compreendeu a seleção das ferramentas e implementações que respeitem os padrões definidos, de forma que se comuniquem bem com os outros componentes. Além disso, com os requisitos de alta disponibilidade e o tratamento de *spam* inteligente, novas variáveis são inseridas no ambiente e que devem ser consideradas no processo de implementação.

Apesar da oferta de várias soluções relacionadas a sistemas de e-mails, é evidente a carência de documentação de qualidade sobre a integração dos componentes e a respeito de alta disponibilidade. Muitas vezes, as únicas informações disponíveis estão em *blogs* ou fóruns da web, além não contar com testes nem garantias dos procedimentos. Além disso, para se ter detalhes de algumas aplicações, se faz necessário consultar o código-fonte, uma vez que a documentação disponível não apresenta todos os detalhes.

Com o objetivo de comprovar as informações dos estudos e assegurar a interoperabilidade dos componentes, um cenário de virtualização foi utilizado para a implementação. Esses recursos foram disponibilizados através da solução VirtualBox (ORACLE, 2013) da Oracle na plataforma GNU/Linux. Várias máquinas virtuais foram utilizadas para os testes e implementações, além de várias versões dos *softwares*.

O objetivo primário do processo de implementação do sistema numa

infraestrutura virtualizada é comprovar a viabilidade do investimento, principalmente no que diz respeito ao seu uso num ambiente de produção. A comunicação entre os componentes e os resultados obtidos a partir daí podem ou não comprovar as premissas agregadas na fundamentação teórica.

Para os testes iniciais, foi criado um *dataset* com mais de 6000 mensagens, categorizadas em *spam* e *ham*. Por se tratar de um ambiente no qual alguns componentes operam em máquinas diferentes, o mecanismo de filtragem deve ser capaz de operar com essas variáveis e oferecer os recursos necessários para os usuários. Além disso, o mecanismo deve ser capaz de garantir esses resultados durante toda a sua operação, possibilitando o treinamento constante do filtro.

O capítulo 4 apresenta as informações sobre o processo de implementação do sistema, como detalhes sobre as aplicações escolhidas e as dificuldades encontradas.

4. IMPLEMENTAÇÃO DO SISTEMA

Após os estudos preliminares e o entendimento de uma possível solução para o problema apresentado, o processo de implementação do sistema seguiu três etapas. A primeira etapa consistiu na estruturação do sistema, na qual foram consideradas as possibilidades para atingir a disponibilidade desejada. A segunda etapa envolveu a implementação do sistema de e-mails na estrutura definida na primeira etapa. Após a conclusão da segunda etapa e o sistema de e-mails estar funcionando perfeitamente com alta disponibilidade, a terceira etapa tratou da implementação do mecanismo de filtragem de *spam*.

4.1. ESTRUTURA DO SISTEMA

O sistema de e-mails aqui proposto foi projetado por partes, tratando individualmente cada um dos elementos. A Figura 6 apresenta um diagrama de alto nível da estrutura proposta para o sistema. Essa estrutura é dividida em duas partes principais: os MTAs e o *mailstore*. O aqui chamado *mailstore* se refere às máquinas que fazem o papel de receber e armazenar as mensagens dos usuários. Além disso, são essas as máquinas que oferecem acesso às caixas postais dos usuários através dos protocolos IMAP e POP3.

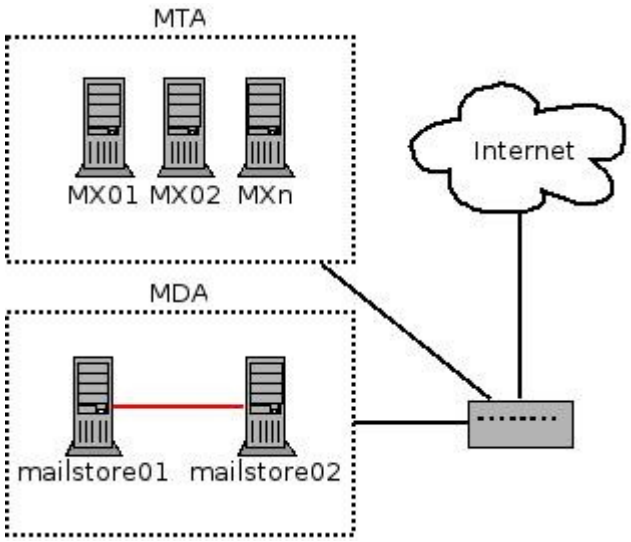

Figura 6: Estrutura do sistema

Como pode ser observado, a organização do sistema tem como base uma estrutura simples, na tentativa de não adicionar maior complexidade à solução. Apesar disso, além da complexidade natural de um sistema de e-mails, as camadas que adicionam alta disponibilidade ao conjunto não é simples.

O diagrama abstrai os detalhes sobre infraestrutura elétrica e de rede, mas para uma aplicação de alta disponibilidade, esses recursos devem ser tratados com a devida atenção. Como a implementação no trabalho foi feita através de virtualização do ambiente, esses detalhes não foram adicionados ao trabalho.

A estrutura apresentada na Figura 6 permite expansão, alterações, manutenção e qualquer tarefa com o sistema, devido principalmente à organização modular. Com a implementação dos componentes de forma isolada, os erros, falhas e vulnerabilidades podem ser identificadas e endereçadas sem comprometer toda a estrutura.

Essa organização tem como principal objetivo oferecer altos índices de disponibilidade, utilizando mecanismos de redundância nos níveis mais críticos. Além disso, o custo do sistema deve ser reduzido, pois poucas máquinas são utilizadas, sem deixar de oferecer a redundância desejada. Essas características e vantagens são tratadas na seção 4.2.

4.2. ALTA DISPONIBILIDADE

Para sustentar a estrutura apresentada, após os vários estudos, a decisão tomada foi favorável ao uso de uma solução baseada na arquitetura GNU/Linux, com os *softwares* e bibliotecas livres, além dos padrões de mercado.

Ao contrário da alta disponibilidade oferecida por serviços proprietários (STANEK, 2009), a proposta deste trabalho é tratar a estrutura em camadas, lidando com cada componente do sistema de forma individual. Essa abordagem permite que as técnicas e conceitos contidos no documento possam ser reutilizada em outras aplicações, da mesma forma que informações de outros trabalhos embasaram esse trabalho.

Os detalhes da implementação, como os códigos e comandos utilizados, foram suprimidos do documento principal, a fim manter o texto enxuto e legível. Essas informações foram adicionadas aos apêndices do documento e contém todos os detalhes da implementação.

As subseções dessa seção apresentam os detalhes sobre a implementação das camadas do sistema.

4.2.1. Rede e conectividade

O primeiro aspecto da alta disponibilidade a ser considerado é a interconexão dos componentes e sua conectividade com a Internet. Cada um dos componentes do sistema de ser conectado à uma rede local, por onde as informações serão transmitidas e o conjunto será controlado.

Em relação à conectividade do MTA, cada um deles deve ser conectado diretamente à rede local com um IP único. Para acessar esses *hosts*, os clientes deverão utilizar nomes, possibilitando assim as configurações de redundância através do DNS. Para as conexões externas, como aquelas de outros sistemas de e-mail para entregar mensagens ao sistema local, são utilizados registros do tipo MX no servidor DNS. Nesse caso, é possível utilizar mais de um registro MX com a mesma prioridade, cada um para um dos servidores, garantindo assim redundância e balanceamento de carga (KLENSIN, 2008).

O recurso adotado para a disponibilidade da conexão no MDA é um *failover* de IP. Esse recurso possibilita que um IP virtual seja negociado entre os nós do *cluster*. Assim, quando o nó com aquele IP virtual falhar, outro nó atribuirá aquele endereço na sua interface de rede e assume o papel de oferecer os recursos aos usuários. Como todos os recursos, exceto o *failover* IP, ficam ativos a todo momento em todos os nós, não há a necessidade de migrar os recursos junto ao *failover* IP.

Esse modelo possibilita alta disponibilidade, porém não fornece balanceamento de carga. Se houver essa necessidade, várias alternativas estão disponíveis, como, por exemplo, um *script* para balanceamento de carga *round-robin* com o iptables, popular *firewall* da plataforma GNU/Linux. Outra alternativa é a implementação do projeto (LINUX VIRTUAL SERVER, 2012), que leva em conta a carga do sistema antes de fazer o redirecionamento das requisições.

Além das conexões normais dos computadores, a estrutura definida requer um canal de comunicação dedicado entre os nós do *cluster*. Essa conexão é utilizada para a replicação dos dados entre os nós e para controle do conjunto. Se houver necessidade, é possível ainda aprimorar esse canal fazendo uma agregação de *links* (LINUX FOUNDATION, 2009).

4.2.2. Solução de armazenamento

No caso do armazenamento, há duas opções viáveis: armazenamento central compartilhado ou armazenamento distribuído. Neste o trabalho, é utilizada uma unidade distribuída de armazenamento, através do DRBD.

Essa solução oferece vantagens e desvantagens. A vantagem mais notável é que os dados permanecem replicados durante todo o tempo e em caso de falha crítica do *hardware* em um dos nós, o outro nó possui os dados atualizados e pode assumir a responsabilidade pelos serviços. É importante não confundir esse recurso com um backup das informações, já que o *cluster* irá sincronizar mesmo as inconsistências do sistema de arquivos.

Outra alternativa é utilizar uma solução de armazenamento especializada, como um NAS ou SAN. Vários modelos estão disponível no mercado, cada um com suas particularidaes e valores. Os modelos EqualLogic PS6500E da DELL é um exemplo de

solução de armazenamento especializada (DELL, 2013b). Nesses casos, a expansão da capacidade de armazenamento é simplificada, possibilitando utilizar vários *terabytes* de discos. Apesar de esse tipo de *hardware* ser extramente especializado e o fornecedor oferecer bons índices de disponibilidade e estabilidade, de certa forma, ela representará um ponto único de falha no sistema. Esse problema pode ser facilmente endereçado ampliando a solução de armazenamento, com o devido investimento financeiro.

Para a implementação do sistema, a solução de armazenamento escolhida utiliza os próprios discos das máquinas. Novamente, é possível pontuar várias vantagens e desvantagens nesse modelo. A Figura 7 mostra um exemplo de servidor de alta capacidade de armazenamento. Mesmo nesse caso, no qual o servidor suporte um número elevado de discos individuais, a capacidade de armazenamento é limitada, quando comparado aos *hardwares* especializados já citados.

Figura 7: Servidor DELL Poweredge C2100 de alta capacidade de armazenamento
(DELL, 2013a)

No caso do servidor da Figura 7, mesmo suportando doze discos individuais e considerando que seja fácil encontrar discos de três *terabytes* no mercado atualmente, esse servidor suportaria trinta e seis *terabytes*. Ao se utilizar uma solução de RAID, como um RAID 1, por exemplo, essa capacidade cai para a metade. Dependendo da necessidade de armazenamento dos usuários, dezoito *terabytes* de dados pode não ser o suficiente para o sistema.

Há ainda a possibilidade de montar o próprio *storage*, já utilizada por algumas

empresas da área (YAN, 2013).

Assim, a escolha desse tipo de solução para o trabalho se justifica ao considerar o uso de mecanismos avançados, como é o caso do DRBD. Se a opção de armazenamento for baseada em armazenamento centralizado, não é necessário a implementação do DRBD, bastando apenas um sistema de arquivo distribuído, como é o caso do OCFS2.

4.2.3. DRBD e OCFS2

Com o DRBD, os discos estão sempre sincronizados e com o sistema de arquivos OCFS2, é possível acessar a unidade de armazenamento simultaneamente nos dois nós. Para que isso seja possível, o recurso DLM deve ser instalado e configurado, como detalhado no Apêndice 1.

O DRBD possui um gerenciador próprio, que cuida da comunicação entre os nós e garante a sincronização dos discos. Porém, como os recursos são gerenciados pelo Pacemaker, pode haver confusão entre as programas. Um problema semelhante é encontrado com os pacotes do OCFS2 para a distribuição. Com a instalação do conjunto de pacotes que fornecem as ferramentas para utilizar o OCFS2, alguns *daemons* são instalados no sistema. Esses *daemons* quando iniciados carregam módulos no *kernel* que causam incompatibilidade com o Pacemaker, causando falhas na iniciação dos recursos. Dessa forma, é necessário desabilitar todos os *daemons* instalados pelos pacotes para que a pilha de "clusterização" correta seja utilizada (DRBD, 2012).

Todos os detalhes da implementação e configurações necessárias estão detalhadas no Apêndice 1.

4.3. SISTEMA DE E-MAILS

Com o *cluster* pronto e a solução de armazenamento operacional, é possível iniciar o processo de instalação dos componentes do sistema de e-mails.

O primeiro componente a ser instalado é o MTA. O Postfix, solução escolhida, é

uma solução de MTA madura, que foi criada com o objetivo principal de substituir o Sendmail. Por esse motivo, o *software* foi inteiramente desenvolvido para ser compatível com concorrente, facilitando o processo de migração. Entre os inúmeros recursos e funcionalidades do Postfix é possível citar os recursos embutidos para controle de *spam*, o amplo suporte para padrões e protocolos do mercado, o suporte a diversas bases de dados, além de outros vários recursos que um bom MTA deve oferecer (DENT, 2003).

Inicialmente, os trabalhos envolveram a solução Cyrus (CYRUS, 2013) para MDA, porém após a identificação de alguns problemas na integração com a solução de armazenamento distribuída, a solução teve que ser substituída. O Cyrus utiliza um mecanismo de armazenamento próprio, que necessita de bloqueio de alguns arquivos para o correto funcionamento. Apesar dessa particularidade ter vantagens (HEINLEIN, 2008), algumas dificuldades são encontradas para os ambientes de múltiplos acessos simultâneos e *backup*, o que obrigou a busca por outra solução.

O Dovecot é um projeto razoavelmente novo que oferece recursos e alguns diferenciais dos concorrentes. A solução foi desenvolvida visando criar mecanismos internos de auto recuperação e auto otimização, garantindo ao sistema continuidade de operação mesmo sob circunstâncias inesperadas. Além disso, o projeto garante a segurança da solução, enquanto também oferece recursos avançados através de *plugins*. O Dovecot tem conformidade com os padrões conhecidos e é compatível com todos os componentes utilizados.

Para contornar o problema com o armazenamento das mensagens no volume distribuído, foi utilizado o formato Mailbox para as mensagens. Dessa forma, o MTA entrega as mensagens através de LMTP para o MDA que armazena as mensagens em arquivos individuais no sistema de arquivos. Por esse motivo, não são necessárias travas de acesso aos arquivos, possibilitando ao *cluster* operar no modo ativo/ativo sem comprometer as caixas postais dos usuários.

Devido à organização do sistema e seus vários componentes, é importante manter uma base centralizada para os dados de usuários e outras informações que serão comuns no sistema. Dessa maneira, o gerenciamento de contas, listas de e-mails, senhas, entre outras tarefas do dia a dia, se tornam mais simples, sendo necessário alterá-las em apenas um lugar. Por exemplo, o gerenciamento de senhas seria um desafio, sendo necessário alterá-la em cada um dos componentes, caso fossem utilizadas contas de usuários nos sistemas de forma

isolada.

E as vantagens não se limitam aos usuários finais e administradores, mas abrangem também todo o sistema. Quando uma nova mensagem chega ao MTA, antes de aceitá-la, é necessário verificar a existência do usuário no banco de dados. Sem esse a base de dados centralizada, os componentes necessitam manter sincronizadas as informações dos usuários, para evitar, por exemplo, a aceitação e encaminhamento de todas as mensagens para verificação no MDA. Ao utilizar a base centralizada, é possível economizar recursos computacionais aceitando apenas as mensagens destinadas aos usuários que realmente existem no sistema. Esses são alguns benefícios de se utilizar uma base centralizada de dados, mas existem ainda muitas outros (RED HAT, 2008).

Para a centralização dos dados de usuário é possível utilizar diversas soluções diferentes, desde um banco de dados até um serviço de diretório. Atualmente, é comum utilizar uma implementação de LDAP (SERMERSHEIM, 2006), que permite ser utilizado não só pelo sistema de e-mails mas por várias outras aplicações numa rede. Apesar de fora do escopo desse trabalho, é importante citar que esse tipo de serviço também deve ser implementado de forma a fornecer altos índices de disponibilidade, já que esse componente passa a ser parte vital do sistema como um todo. Existem trabalhos na área de alta disponibilidade com LDAP que oferecem boas referências e instruções para a implementação (HAFERKAMP, 2011).

4.4. FILTRAGEM DE SPAM

Com o sistema de e-mails completo e em funcionamento, resta concluir a implementação do mecanismo de filtragem de *spam*. Apesar de o Capítulo 2 apresentar diversos mecanismos de filtragem de *spam*, a fase de implementação não visa detalhar todos eles. As informações sobre outros mecanismos para mitigação de *spam* foram incluídas no trabalho com o objetivo de apresentar a diversidade de técnicas utilizadas para essa tarefa. O objetivo do trabalho é tratar a implementação do mecanismo de filtragem inteligente, já que os outros mecanismos são maduros e amplamente difundidos na Internet.

Existem hoje várias soluções livres para o tratamento de *spam*. Algumas delas implementam mais de uma técnica de filtragem, buscando aprimorar os resultados. No caso

desse trabalho, após entender os mecanismos de filtragem, se fez necessário encontrar uma solução com uma implementação mais pura, que representasse a técnica documentado na literatura. Por esse motivo, o Bogofilter foi selecionado.

O Bogofilter (BOGOFILTER, 2013) é baseado no trabalho de (GRAHAM, 2003a), além de contar com algumas otimizações no algoritmo (GRAHAM, 2003b) (ROBINSON, 2003). A solução é escrita em linguagem C e apresenta bom desempenho nas operações. Para avaliar seu funcionamento e a acuracidade dos resultados, um *dataset* com pouco mais de seis mil e-mails foi criado, sendo constituído de mensagens classificadas como *spam* e mensagens classificadas como *ham*.

Essa solução possibilita ainda manter bancos de dados individuais por usuários, a fim de oferecer resultados diferenciados. Dessa forma, os resultados tendem a ser mais precisos, já que as mensagens pessoais terão maior influência na base de dados individuais, consequentemente nos resultados das filtragens (GRAHAM, 2003c).

As mensagens do *dataset* que originalmente foram organizadas no formato *Maildir*, tinham como nome dos arquivos o assunto da mensagem e a extensão "eml". Essas mensagens foram renomeadas para o *hash* MD5 de seu conteúdo e foram separadas aleatoriamente em dez grupos. Esses grupos de mensagens foram utilizadas para treinamento cruzado do Bogofilter, ou seja, todas os grupos foram utilizados para treinar e testar a acuracidade da solução.

Com o *dataset* criado, a solução Bogofilter apresentou taxa de incerteza para detecção de *ham* de aproximadamente 0.42% e para detecção de *spam* de aproximadamente 4.58%, superiores aos testes realizados com filtros de regras estáticas. Como o filtro necessita de treinamento, o procedimento inicial pode não ser suficiente para bons resultados. Nesses casos, é necessário treinar o mecanismo com mais mensagens ou apenas aguardar que o treinamento continuado forneça estatísticas suficientes para isso.

Para garantir que o banco de dados de *spam* e *ham* continue sempre atualizado e ativo, o sistema conta com um mecanismo para treinamento contínuo do filtro. Quando um usuário move uma mensagem para a pasta de *spam*, o sistema informa o filtro que utiliza o conteúdo da mensagem para aprimorar a base de dados, também conhecida como *bag-of-words*.

Assim, a tarefa de filtrar de e-mails é distribuída pelo sistema. Nos *mailstores* o Bogofilter interage com o plugin *anti-spam* do Dovecot (DOVECOT, 2013) e trabalha a base

de dados com novas palavras e informações. Essa base de dados é, de acordo com o programado, transferida para os MTA, onde acontece efetivamente a filtragem.

O diagrama da Figura 8 apresenta o modelo de operação do filtro *anti-spam*. O filtro é treinado nos *mailstores* através da interação dos usuários e essa base de dados é utilizada na filtragem nos MTA. Quando um usuário move uma mensagem para a pasta Spam, o plugin *anti-spam* do Dovecot executa uma instância do Bogofilter para que o mesmo treine o mecanismo com aquela mensagem. O mesmo acontece quando uma mensagem é movida da pasta *spam* para outra pasta do sistema.

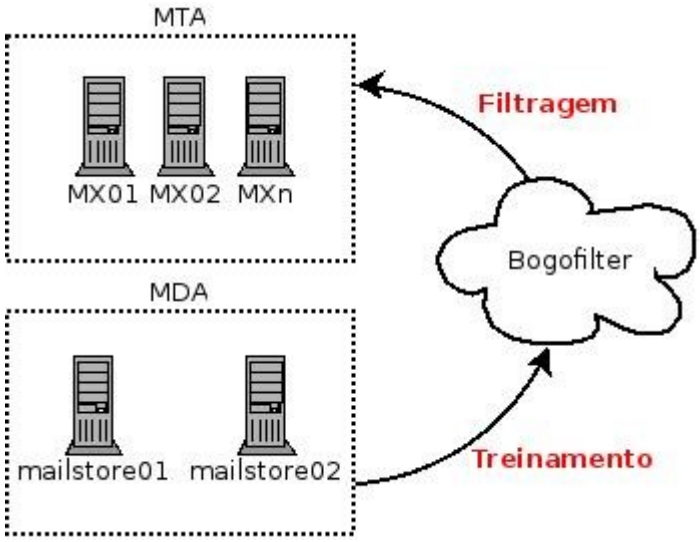

Figura 8: Modelo de operação do Bogofilter

Além do treinamento executado quando mensagens são movidas pelos usuários nas pastas da caixa postal, um outro treinamento pode ser executado. Quando uma mensagem chega ao MTA e o filtro a classifica como *spam*, é possível que nesse mesmo momento o filtro utilize o conteúdo dessa mensagem já classificada para o treinamento. No entanto, se essa classificação ocorreu de forma equivocada, o filtro pode ser influenciado de forma inesperada.

Para contornar esse problema, o treinamento no MTA é ativado no sistema apresentado. Ao contrário, quando uma mensagem é classificada como *spam*, um campo é adicionado ao seu cabeçalho indicando a classificação. Quando essa mensagem chega ao *mailstore*, ela passa por um filtro Sieve que a direciona para a pasta *spam*. Nesse momento, o

sistema utiliza o conteúdo da mensagem para treinar e aprimorar a *bag-of-words*. Caso a classificação tenha sido equivocada, o usuário detectará o problema e moverá a mensagem para outra pasta. Nesse momento, o Bogofilter será informado da mudança, treinando o sistema novamente o compensando o erro.

As instruções sobre a implementação desses componentes são bem detalhadas nos sites oficiais dos projetos, sendo necessários apenas preencher as variáveis para a integração dos componentes. A transmissão do banco de dados de *spam* entre o *mailstore* e o MTA é feita através de um *script* simples em liguagem Shellscript e é executado pelo Cron (agendador de tarefas do Linux) num período determinado.

Com a organização e modo de operação proposta, o usuário tem a sua disposição um mecanismo eficiente de filtragem de *spam*, o qual pode influenciar movendo as mensagens entre as pastas de sua caixa postal. Esse mecanismo é semelhante ao oferecido pelos grandes provedores e deve atender as expectativas dos usuários.

5. CONCLUSÃO

Com o desenvolvimento do trabalho apresentado, foi possível entender o que é um sistema de e-mails em detalhes, além das dificuldades com o tratamento de *spam*. Manter um sistema com altos índices de disponibilidade é, de fato, um desafio, porém não é impossível. Com a grande oferta de componentes e soluções para esse tipo de sistema, o grande desafio passa a ser entender e integrar as peças do grande quebra-cabeça.

Após os estudos, testes e implementações constatou-se que a estrutura organizada é capaz de atender as necessidades destacadas no início do trabalho, de forma que os usuários tenham recursos avançados a sua disposição.

Novamente, o *software* livre mostrou sua flexibilidade e o grande poder agregado à filosofia de compartilhar os avanços e esforços, garantindo que a comunidade avance de maneira efetiva e sem desigualdade.

Quanto aos filtros de *spam* baseados em aprendizado, os resultados alcançados são superiores aos filtros com regras estáticas, porém há uma maior necessidade de recursos computacionais. Com a possibilidade do treinamento permanente do mecanismo, os usuários podem influenciar nos resultados, possibilitando que suas necessidades sejam atendidas.

Dessa maneira, fica constatada a viabilidade do sistema, caracterizando uma solução completa para um ambiente de produção, como nos casos das instituições que não podem utilizar serviços de terceiros e necessitam de um sistema seguro e livre de *spam*.

6. REFERÊNCIAS BIBLIOGRÁFICAS

ANTISPAM.BR. *Gerência da Porta 25 - Antispam.br*. Disponível em: <http://antispam.br/admin/porta25/>. Acesso em: 13 maio 2013.

BLANZIERI, E.; BRYL, A. A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review*, v. 29, n. 1, p. 63–92, 2008.

BOGOFILTER. *Bogofilter Home Page*. Disponível em: <http://bogofilter.sourceforge.net/>. Acesso em: 15 maio 2013.

BOOTH. *booth*. Disponível em: <https://github.com/ClusterLabs/booth>. Acesso em: 12 maio 2013.

CARTER, D.; FINCH, T. Scaling up Cambridge University's email service. 2004, [S.l: s.n.], 2004.

CLUSTER LABS. *Cluster Labs - The Home of Pacemaker*. Disponível em: <http://clusterlabs.org/>. Acesso em: 13 maio 2013.

CLUSTER LABS. *Clusters from Scratch*. Disponível em: <http://clusterlabs.org/doc/en-US/Pacemaker/1.1-crmsh/html-single/Clusters_from_Scratch/index.html>. Acesso em: 13 maio 2013a.

CLUSTER LABS. *ClusterTypes - ClusterLabs*. Disponível em: <http://clusterlabs.org/wiki/ClusterTypes>. Acesso em: 10 maio 2013b.

COROSYNC. *Corosync by corosync*. Disponível em: <http://corosync.github.io/corosync/>. Acesso em: 11 maio 2013.

CRISPIN, M. *INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1*. [S.l.]: IETF, 2003. Disponível em: <http://www.ietf.org/rfc/rfc3501.txt>. (Request for Comments, 3501).

CYRUS. *Project Cyrus*. Disponível em: <http://www.cyrusimap.org/>. Acesso em: 15 maio 2013.

DAKE, S. C.; CAULFIELD, C.; BEEKHOF, A. The corosync cluster engine. 2008, [S.l: s.n.], 2008.

DELL. *Detalhes do servidor em rack PowerEdge C2100 | Dell Brasil*. Disponível em: <https://www.dell.com/br/corporativo/p/poweredge-c2100/pd>. Acesso em: 10 maio 2013a.

DELL. *SAN iSCSI Dell EqualLogic PS6500E | Dell Brasil*. Disponível em: <https://www.dell.com/br/corporativo/p/equallogic-ps6500e/pd>. Acesso em: 11 maio 2013b.

DENT, K. *Postfix: The Definitive Guide*. 1st. ed. [S.l.]: O'Reilly Media, 2003.

DOVECOT. *Plugins/Antispam - Dovecot Wiki*. Disponível em: <http://wiki2.dovecot.org/Plugins/Antispam>. Acesso em: 15 maio 2013.

DRBD. *DRBD:What is DRBD*. Disponível em: <http://www.drbd.org/>. Acesso em: 14 maio 2013.

DRBD. *Pacemaker OCFS2 management.* Disponível em: <http://www.drbd.org/users-guide/s-ocfs2-pacemaker.html>. Acesso em: 15 maio 2013.

FASHEH, M. OCFS2: The Oracle clustered file system, version 2. 2006, [S.l: s.n.], 2006. p. 289–302.

GANDINI, J. A. D.; SALOMÃO, D. P. DA S.; JACOB, C. A validade jurídica dos documentos digitais. *Jus Navigandi*, 1 ago. 2002. Disponível em: <http://jus.com.br/revista/texto/3165>. Acesso em: 13 maio 2013.

GRAHAM, P. A plan for spam, 2002. 2003a, [S.l: s.n.], 2003.

GRAHAM, P. Better bayesian filtering. 2003b, [S.l: s.n.], 2003. p. 15–17.

GRAHAM, P. Will Filters Kill Spam? *Computer Security Journal*, 2003c.

HAAS, F. Ahead of the pack: the pacemaker high-availability stack. *Linux J.*, v. 2012, n. 216, abr. 2012a. Disponível em: <http://dl.acm.org/citation.cfm?id=2208859.2208863>.

HAAS, F. Replicate everything! highly available iSCSI storage with DRBD and pacemaker. *Linux J.*, v. 2012, n. 217, maio 2012b. Disponível em: <http://dl.acm.org/citation.cfm?id=2240076.2240081>.

HAFERKAMP, R. OpenLDAP in High Availability Environments. 2011.

HANSEN, T.; CROCKER, D.; HALLAM-BAKER, P. *DomainKeys Identified Mail (DKIM) Service Overview*. [S.l.]: IETF, 2009. Disponível em: <http://www.ietf.org/rfc/rfc5585.txt>. (Request for Comments, 5585).

HEINLEIN, P. *The book of IMAP: building a mail server with Courier and Cyrus*. U.S. ed ed. Munich : San Francisco: Open Source Press ; No Starch Press, 2008.

KLENSIN, J. *Simple Mail Transfer Protocol*. [S.l.]: IETF, 2008. Disponível em: <http://www.ietf.org/rfc/rfc5321.txt>. (Request for Comments, 5321).

KÖVI, A.; VARRÓ, D.; NÉMETH, Z. Making legacy services highly available with OpenAIS: an experience report. ISAS'06, 2006, Berlin, Heidelberg. *Anais*... Berlin, Heidelberg: Springer-Verlag, 2006. p. 206–216. Disponível em: <http://dx.doi.org/10.1007/11955498_15>.

KUCHERAWY, M.; CROCKER, D. *Email Greylisting: An Applicability Statement for SMTP*. [S.l.]: IETF, 2012. Disponível em: <http://www.ietf.org/rfc/rfc6647.txt>. (Request for Comments, 6647).

LINUX FOUNDATION. *bonding | The Linux Foundation*. Disponível em: <http://www.linuxfoundation.org/collaborate/workgroups/networking/bonding>. Acesso em: 15 maio 2013.

LINUX-HA. *Linux-HA*. Disponível em: <http://www.linux-ha.org/wiki/Main_Page>. Acesso em: 17 maio 2013.

LINUX VIRTUAL SERVER. *The Linux Virtual Server Project - Linux Server Cluster for Load Balancing*. Disponível em: <http://www.linuxvirtualserver.org/>. Acesso em: 15 maio 2013.

MYERS, J. *Local Mail Transfer Protocol*. [S.l.]: IETF, 1996. Disponível em: <http://www.ietf.org/rfc/rfc2033.txt>. (Request for Comments, 2033).

MYERS, J.; ROSE, M. *Post Office Protocol - Version 3*. [S.l.]: IETF, 1996. Disponível em: <http://www.ietf.org/rfc/rfc1939.txt>. (Request for Comments, 1939).

OPENAIS. *OpenAIS*. Disponível em: <http://freecode.com/projects/openais>. Acesso em: 14 maio 2013.

ORACLE. *Oracle VM VirtualBox*. Disponível em: <https://www.virtualbox.org/>. Acesso em: 14 maio 2013.

PERRIN, C. *The CIA Triad | TechRepublic*. Disponível em: <http://www.techrepublic.com/blog/security/the-cia-triad/488>. Acesso em: 14 maio 2013.

PROJECT OCFS2. *Project: OCFS2*. Disponível em: <https://oss.oracle.com/projects/ocfs2/>. Acesso em: 11 maio 2013.

RED HAT. *Why Use LDAP?* Disponível em:

<https://www.centos.org/docs/5/html/5.2/Deployment_Guide/s1-ldap-adv.html>. Acesso em: 14 maio 2013.

ROBINSON, G. *A Statistical Approach to the Spam Problem | Linux Journal*. Disponível em: <http://www.linuxjournal.com/article/6467>. Acesso em: 15 maio 2013.

SECURITY SPACE. *Mail (MX) Server Survey*. Disponível em: <http://www.securityspace.com/s_survey/data/man.201204/mxsurvey.html>. Acesso em: 14 maio 2013.

SERMERSHEIM, J. *Lightweight Directory Access Protocol (LDAP): The Protocol*. [S.l.]: IETF, 2006. Disponível em: <http://www.ietf.org/rfc/rfc4511.txt>. (Request for Comments, 4511).

SIMPSON, K.; BEKMAN, S. *Fingerprinting the World's Mail Servers - O'Reilly Media*. Disponível em: <http://www.oreillynet.com/pub/a/sysadmin/2007/01/05/fingerprinting-mail-servers.html>. Acesso em: 10 maio 2013.

SPAMHAUS. *The Spamhaus Project - The Definition of Spam*. Disponível em: <http://www.spamhaus.org/consumer/definition/>. Acesso em: 11 maio 2013a.

SPAMHAUS. *The Spamhaus Whitelist*. Disponível em: <http://www.spamhauswhitelist.com/en/about.html>. Acesso em: 13 maio 2013b.

STANEK, W. R. *Ensuring High Availability in Microsoft Exchange Server 2010 - TechNet Magazine Article - December 2009*. Disponível em: <http://technet.microsoft.com/en-us/magazine/ee835711.aspx>. Acesso em: 12 maio 2013.

SYMANTEC. *2013 Internet Security Threat Report*. Threat Report, nº Volume 18. [S.l.]: Symantec Coporation, 2013. Disponível em: <http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v18_2012_21291018.en-us.pdf'>. Acesso em: 11 maio 2013.

WONG, M.; SCHLITT, W. *Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1*. [S.l.]: IETF, 2006. Disponível em: <http://www.ietf.org/rfc/rfc4408.txt>. (Request for Comments, 4408).

YAN, M. *Cold Storage Hardware v0.5*. . [S.l.]: Open Compute Project. Disponível em: <http://www.opencompute.org/wp/wp-content/uploads/2013/01/Open_Compute_Project_Cold_Storage_Specification_v0.5.pdf >. , 2013

7. APÊNDICE 1 – ROTEIRO DE CONFIGURAÇÃO DO CLUSTER

O roteiro deste documento apresenta as etapas a serem seguidas para a configuração de um *cluster* no Linux para o *mailstore*. Essas instruções são válidas para o sistema operacional GNU/Linux Debian Wheezy, que foi a distribuição utilizada ao decorrer do trabalho. As instruções podem variar de acordo com as particularidades do sistema instalado. Além disso, as configurações aqui apresentadas foram utilizadas para a configuração de um *cluster* de duas máquinas, sendo que ao utilizar mais computadores, podem haver pequenas alterações nas informações.

Para iniciar as configurações, assume-se que as máquinas estejam como o sistema operacional instalado, atualizado e configurado corretamente. A fim de manter o roteiro simples e direto, essas configurações não estão inclusas, mas podem ser facilmente encontradas na documentação oficial do projeto. Outro requisito importante é a infraestrutura de rede. Para o uso do DRBD e dos recursos de *cluster* em geral, é viável possuir uma conexão dedicada entre os servidores para a troca de informações referentes a essas aplicações.

Após a preparação inicial, se inicia a configuração dos servidores para o uso do *cluster*. O primeiro passo é instalar os pacotes e dependências necessárias. O comando a seguir instala todos os pacotes básicos que serão utlizados e o sistema de empacotamento do Debian se encarrega de instalar corretamente as dependências:

```
# aptitude install corosync pacemaker drbd8-utils
ntp ocfs2-tools-pacemaker dlm-pcmk openais
chkconfig
```

Com os pacotes instalados, pode-se iniciar a configuração dos mesmos. Um requisito importante para esse tipo de *cluster*, é a sincronização de tempo dos seus membros. Uma solução é utilizar um protocolo de sincronização como o NTP através do pacote de mesmo nome. Um exemplo de configuração é o arquivo `/etc/ntp.conf` possuir a informação abaixo.

```
server pool.ntp.br
```

Vale lembrar que se houver um servidor NTP mais próximo, talvez seja melhor utilizá-lo. Quanto mais rápida a comunicação com servidor, melhores os resultados nas sincronizações de tempo.

Após essa etapa, se inicia a configuração da unidade de armazenamento através do DRBD. Para tal, os arquivos a seguir dever ser editados:

```
# nano /etc/drbd.d/global_common.conf
# nano /etc/drbd.d/mailstore.res
```

Esses arquivos mantém as informações do DRBD sobre o recurso a ser utilizado. O exemplo abaixo representa uma configuração válida do recurso mailstore.

```
Adicionar conteúdo do arquivo mailstore.res
```

Como o recurso deverá operar no modo ativo/ativo, convém-se configurar algum mecanismo de *fencing*, que entrará em ação quando o link entre os nós for interrompido. A configuração abaixo representa uma alternativa de configuração utilizando os arquivos providos pelo próprio DRBD.

```
  disk {
        fencing resource-and-stonith;
  }

  handlers {
        fence-peer
"/usr/lib/drbd/crm-fence-peer.sh";
        after-resync-target
"/usr/lib/drbd/crm-unfence-peer.sh";
  }
```

Para criar os metadados do dispositivo, o comando a seguir é utilizado. O argumento a ser informado é o nome do recurso a ser iniciado. Esses comandos precisam ser excutados em cada um dos nós.

```
# drbdadm create-md mailstore
# drbdadm up mailstore
```

A partir dessa etapa, para forçar a sincronização inicial, um dos nós devem ser escalado como primário, inciando assim a sincronização.

```
# drbdadm primary --force mailstore
```

A configuração do Corosync se resume apenas em alterar o arquivo `/etc/corosync/corosync.conf` e alterar o endereço através do qual os nós irão se comunicar. Para concluir, a execução do comando `corosync-keygen` gera a chave criptográfica, que deverá ser copiada para o outros nós do *cluster*.

Durante a configuração do *cluster* várias dificuldades foram encontradas. Essa versão do Debian, talvez por ser nova, tem algumas incoerências no que se diz respeito à configuração do *cluster*. De uma forma resumida, os comandos a seguir devem resolver o problema e possibilitar a conclusão da configuração. Eles desabilitam o os *daemon* `ocfs2` e `o2cb` de serem carregados no boot e evitam que alguns módulos incompatíveis sejam carregados.

```
# chkconfig -d ocfs2
# chkconfig -d o2cb
```

Por se tratar de um *cluster* com apenas dois nós, quando um deles não estiver operacional, o outro nó não terá quórum para assumir os recursos. Assim, para desabilitar a poítica de quórum utiliza-se o comando a seguir no `crm`:

```
crm # configure property no-quorum-policy=ignore
```

Além disso, como o Pacemaker será encarregado de gerenciar o DRBD, o *deamon* `drbd` deve ser desativado.

```
bash # chkconfig -d drbd
```

O primeiro recurso a ser configurado é o *failover* IP. Esse recurso é responsável por manter gerenciar o endereço IP destinado ao acesso externo ao *cluster*.

```
crm # configure primitive p_ip_failover
ocf:heartbeat:IPaddr2 \
params ip=10.0.0.60 cidr_netmask=32 nic=eth1 \
op monitor interval=30s
```

Para incluir o recurso do DRBD, os comandos a seguir são utilizados:

```
crm # primitive p_drbd_ocfs2 ocf:linbit:drbd \
  params drbd_resource="mailstore"
crm # ms ms_drbd_ocfs2 p_drbd_ocfs2 \
  meta master-max=2 clone-max=2 notify=true
```

Esse recursos possuem parâmetros avançados de configuração que pode ser requisitos para algumas aplicações.

O DLM é executado no *cluster* através do agente `controld`. Para utilizá-lo, é necessário configurar o Corosync para ele usar os recursos do OpenAIS. Adicionar o conteúdo a seguir no arquivo `/etc/default/corosync` deve atender essa necessidade.

```
export COROSYNC_DEFAULT_CONFIG_IFACE= \
"openaisserviceenableexperimental:corosync_parser"
```

Após reiniciar o *daemon* do Corosync, é possível adicionar o recurso controld ao Pacemaker.

```
crm # primitive p_controld ocf:pacemaker:controld
```

O recurso para gerenciamento do sistema de arquivos OCFS2 deve ser adicionado ao Pacemaker. Além dele, configurações são necessárias para agrupar a execução do OCFS2 ao DLM.

```
crm # primitive p_o2cb ocf:pacemaker:o2cb
crm # group g_ocfs2mgmt p_controld p_o2cb
crm # clone cl_ocfs2mgmt g_ocfs2mgmt meta
interleave=true
```

Nesse momento, é importante verificar o *cluster stack* em uso. Essa informação é mantida em tempo de execução no arquivo `/sys/fs/ocfs2/cluster_stack`, que deve conter o valor "`pcmk`".

Após adicionar o recurso do sistema de arquivos, o *cluster stack* deverá ser alterado. Os camando a seguir adicionam o recurso ao Pacemaker.

```
crm # primitive p_fs_ocfs2
ocf:heartbeat:Filesystem \
    params device="/dev/drbd/by-res/mailstore" \
    directory="/mailstore" \
    fstype="ocfs2" options="rw,noatime"
crm # clone cl_fs_ocfs2 p_fs_ocfs2
```

Nesse momento, uma incompatibilidade foi encontrada com *cluster stack*. Quando se utiliza a ferramenta `mkfs.ocfs2` ela munda a *stack* do sistema o que depois inviabiliza o uso com o Pacemaker. Por isso, é necessário alterar a *stack* utilizada na criação do sistema de arquivos, ação que pode ser executada com o comando `tunefs.ocfs2`.

```
# tunefs.ocfs2 --update-cluster-stack /dev/drbd1
```

O comando `debugfs.ocfs2 /dev/drbd1` pode ser utilizado para verificar a *stack* do sistema de arquivos. É necessário que ambos o sistema de arquivos e o sistema operacional estejam configurados para usar a mesma *stack* de clusterização, a `pcmk`.

Com a conclusão dessas configurações, resta apenas adicionar as informações sobre o relacionamento e dependências entre os recursos.

```
crm # order o_ocfs2 inf: ms_drbd_ocfs2:promote
    cl_ocfs2mgmt:start cl_fs_ocfs2:start
crm # colocation c_ocfs2 inf: cl_fs_ocfs2
cl_ocfs2mgmt ms_drbd_ocfs2:Master
```

Essas regras garantem que quando um recurso seja ativado, suas dependências também estejam no mesmo nó. Nesse momento, o *cluster* deve estar operacional e o comando `crm_mon` deve apresentar o status dos serviços configurados. O apêndice 2 apresenta essas configurações de forma unificada.

8. APÊNDICE 2 – ARQUIVO DE CONFIGURAÇÃO DO PACEMAKER

Este documento contém as configurações unificadas do Pacemaker.

```
node mailstore01
node mailstore02
primitive ClusterIP ocf:heartbeat:IPaddr2 \
    params ip="10.0.0.60" cidr_netmask="24"
nic="eth1" \
    op monitor interval="5s" \
    meta target-role="Started" is-managed="true"
primitive p_controld ocf:pacemaker:controld
primitive p_drbd_ocfs2 ocf:linbit:drbd \
    params drbd_resource="mailstore"
primitive p_o2cb ocf:pacemaker:o2cb \
    meta target-role="Started"
group g_ocfs2mgmt p_controld p_o2cb
ms ms_drbd_ocfs2 p_drbd_ocfs2 \
    meta master-max="2" clone-max="2"
notify="true" target-role="Started"
clone cl_ocfs2mgmt g_ocfs2mgmt \
    meta interleave="true"
location cli-prefer-ClusterIP ClusterIP \
    rule $id="cli-prefer-rule-ClusterIP" inf:
#uname eq mailstore01
property $id="cib-bootstrap-options" \
```

```
    dc-version="1.1.7-
ee0730e13d124c3d58f00016c3376a1de5323cff" \
    cluster-infrastructure="openais" \
    expected-quorum-votes="2" \
    stonith-enable="false" \
    last-lrm-refresh="1367975563" \
    stonith-enabled="false" \
    no-quorum-policy="ignore"
rsc_defaults $id="rsc-options" \
    resource-stickiness="1000"
```