Centro Universitário de Brasília

Instituto CEUB de Pesquisa e Desenvolvimento - ICPD

MARIO SERGIO KIRDEIKA JUNIOR

ALTA DISPONIBILIDADE DE RECURSOS EM REDES CORPORATIVAS

Brasília
2015

MARIO SERGIO KIRDEIKA JUNIOR

ALTA DISPONIBILIDADE DE RECURSOS EM REDES CORPORATIVAS

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB/ICPD) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Pós-graduação Lato Sensu, na área de Redes com Ênfase em Segurança.

Orientador: Prof. MsC Francisco Javier De Obaldía Díaz.

Brasília
2015

MARIO SERGIO KIRDEIKA JUNIOR

ALTA DISPONIBILIDADE DE RECURSOS EM REDES CORPORATIVAS

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB/ICPD) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Pós-graduação Lato Sensu, na área de Redes com Ênfase em Segurança.

Orientador: Prof. MsC Francisco Javier De Obaldía Díaz.

Brasília, 09 de outubro de 2015

Banca examinadora

_____

Prof. Dr. Gilson Ciarallo

_____

Prof. Ms. Rafael Sarres de Almeida

RESUMO

O presente trabalho propôs apresentar um método para solucionar um problema de disponibilidade de recursos básicos em redes corporativas: a entrega de configurações de endereços, rotas, autenticação e resolução de nomes. Foi premissa o uso de *softwares* livres para permitir que a partir de uma única e replicável fonte de informações fosse possível fornecer tais serviços. Ainda, esta fonte assim como todos os serviços deveriam ter associadas técnicas de alta disponibilidade, associadas a monitoramento do seu funcionamento e *backup* automatizado e criptografado. Deveriam ser analisadas as alternativas de *software* disponíveis, especificados requisitos, instalado e testado um sistema que entregasse o requisitado. Também, deveria ser criada toda a documentação do que fosse realizado, na forma de roteiros que possibilitassem a replicação da solução, deveriam ser implantados meios de replicação dos dados, testes automatizados, realizadas simulações de falhas e analisadas as respostas do sistema, com ensaios de desempenho e a implantação de uma interface amigável para edição dos dados do sistema. Foi proposto que os objetivos seriam atingidos usando técnicas de virtualização e a inteligência do sistema seria implementada através de *scripts*. Todos os objetivos foram atingidos, com a criação de políticas e as consequentes inteligências de sistema construídas e documentadas, notadamente nos apêndices. Foi conclusão deste trabalho a plena possibilidade de execução deste sistema usando apenas *softwares* livres, sua real implementação usando virtualização e os testes foram realizados com sucesso. Foi também conclusão deste trabalho a nova e real necessidade de que administradores de sistemas devem ter conhecimentos de desenvolvimento de *software*, o que em muito auxilia suas tarefas em vista da tendência da área de Tecnologia da Informação associar as áreas de desenvolvimento e infraestrutura a metodologias ágeis, com plena automação dos processos, em culturas como serviços em nuvem e *DevOps*.

Palavras-chave: *Linux*, Alta disponibilidade, redundância, replicação, *OpenLDAP*.

ABSTRACT

This study undertook to propose a method to solve a problem of availability of basic resources on corporate networks, the delivery address configuration, routing, authentication and name resolution. It was premised on the use of free *software* to allow from a unique and replicable source of information was possible to provide such services. Still, this source as well as all services should have associated high availability techniques, associated monitoring of its operation and automated and encrypted *backup*. Should be analyzed *software* alternatives available, specified requirements, installed and tested a system that delivers the requested. Also should be established all documentation of what was done in the form of *scripts* that would enable replication of the solution, data replication means should be deployed, automated testing, conducted crash simulations and analyzed system responses, with trials performance and the implementation of a user-friendly interface for editing system data. It was proposed that the objectives would be achieved using virtualization techniques and system intelligence would be implemented through *scripts*. All objectives have been achieved with the creation of policies and the resulting constructed system of intelligence and documented, notably in the appendices. Completion of this work was full enforceability of this system using only free *software*, its actual implementation using virtualization and the tests were carried out successfully. It was also the conclusion of this work the new real need for system administrators must have *software* development skills, which greatly assists their tasks in view of the trend in the Information Technology area associate the areas of development and infrastructure to Agile with full automation of the processes in crops such as cloud services and DevOps.

Keywords: High availability, *Linux*, redundance, replication, *OpenLDAP*.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

SUMÁRIO

INTRODUÇÃO

Recursos básicos disponibilizados aos clientes de uma rede de computadores, como fornecimento de endereços, resolução de nomes, autenticação e rotas de acesso a recursos e serviços, são essenciais ao funcionamento de qualquer dispositivo conectado a esta.

Em ambiente corporativo, este fornecimento de recursos é crítico, pois implica diretamente no funcionamento dos dispositivos interligados, e consequentemente impacta o fluxo do trabalho relacionado a estes, sejam eles estações de trabalho, impressoras, relógios de ponto, *tablets*, *smartphones*, etc.

Assim, propiciar meios de que tais dispositivos tenham sempre que necessário o fornecimento de tais informações de acesso é primordial à manutenção do fluxo do trabalho, cujas interrupções são prejudiciais a ponto de causarem prejuízos financeiros.

A auditabilidade dos *softwares* fornecidos deve ser possível, pois são conhecidas formas de vigilância entre empresas, como frequentemente reportadas pela imprensa, haja visto casos de espionagem industrial e até entre países, como o ocorrido com a espionagem de *e-mails* da presidência pelo governo norte americano[1].

O Problema

Como, a partir de uma fonte de informações única e replicável, fornecer endereços de rede, rotas de acesso, resolução de nomes e autenticação dos usuários, sem interrupções, usando apenas *softwares* livres ?

1 - Fonte: Documentos da NSA apontam Dilma Rousseff como alvo de espionagem. Portal G1. Disponível em: <http://g1.globo.com/politica/noticia/2013/09/documentos-da-nsa-apontam-dilma-rousseff-como-alvo-de-espionagem.html>. Acesso em: 14 set. 2015.

Como estabelecer redundância para o serviço oferecido por estes programas, assim como para uma fonte de informações consistente, disponível, confidencial e replicável? Em caso de falha total do equipamento no qual estariam instalados estes programas, o serviço deveria continuar a funcionar, sem interrupções ou prejuízos para os diversos setores da empresa.

Como realizar o *backup* automático e confidencial destas informações, de modo a que se possa rapidamente restabelecer ou até mesmo reconstruir tais serviços, a partir de um *backup* consistente?

Justificativa

A necessidade de baixo uso de recursos, de alta disponibilidade e segurança, flexibilidade na implantação, na manutenção e na reconfiguração são imperativos em sistemas deste tipo.

A não dependência de um único fornecedor, o fomento do desenvolvimento de conhecimento e *software*, e a possibilidade de replicação indefinida da solução justificam plenamente o uso de *software* livre para desenvolver soluções deste tipo, e ainda há a possibilidade de contratação de suporte especializado e de auditabilidade imediata, verificando-se e atestando-se que os *softwares* envolvidos não estejam preparados ou alterados para que seja possível a espionagem dos recursos da empresa.

Objetivo Geral

- Analisar as alternativas de *software*, especificar requisitos destes, instalar e testar um sistema que forneça endereços de rede, rotas de acesso, resolução de nomes e autenticação a redes IPv4, que suporte falha num ou mais componentes da solução, sem prejuízo para as máquinas clientes deste sistema, que permita o acesso e alterações aos dados de forma segura e

faça replicação e *backup* de forma automática e criptografada, para um universo de clientes restrito a um ou mais locais, interligados por redes próximas ou de baixa latência.

Objetivos específicos

- Criar um roteiro que possibilite a instalação e verificação do funcionamento do sistema integralmente;
- Implantar meios de sincronizar as fontes de informações do sistema, de forma rápida e confidencial;
- Realizar *backup* destes dados, de forma periódica, automática, e em local disponível e de forma confidencial e confiável;
- Testar a disponibilidade dos serviços de forma automatizada, periódica e consistente;
- Realizar testes simulando falhas para avaliar o comportamento das políticas, algoritmos e *scripts* construídos;
- Realizar ensaios de desempenho;
- Avisar o administrador do sistema, com detalhes, em caso de falha;
- Implementar uma interface amigável e de fácil acesso, autenticada e criptografada, para que seja possível alterar dados das fontes de informações do sistema.

Procedimentos Metodológicos

O principal objetivo a ser atingido é a definição de requisitos e *softwares*, instalação e testes de um sistema composto por ao menos duas máquinas, cada qual com seu endereço único de rede. Nestas, sob eleição devidamente controlada dependente do estado dos serviços, como fonte de informações, resolução de nomes e fornecimento automático de endereços de rede, haverá um único endereço de rede a mais respondendo por tais serviços, em apenas uma interface de rede, e este endereço deverá ser entregue aos dispositivos clientes como fonte de tais serviços de rede.

Como serão instalados diversos servidores, um ambiente onde estas máquinas possam ser virtualizadas revela-se ideal. Tanto pela facilidade de cópia de máquinas virtuais, o que facilita e agiliza o processo de replicação das instalações, como igualmente facilita o *backup* e exportação da solução de forma integral.

Uma avaliação inicial dos sistemas operacionais e *softwares* existentes que serviriam aos objetivos leva a conclusão que praticamente qualquer distribuição do sistema operacional *Linux*, em suas mais diversas e recentes variações, poderia ser utilizada. Devido à familiaridade do autor com a distribuição *ArchLinux*, pelos métodos simplificados e não orientados a interfaces de configuração, esta distribuição fica selecionada como escolha inicial. Pela facilidade de implementação e disponibilidade, o ambiente *VirtualBox* será utilizado para criar o ambiente virtual.

Será realizada uma pesquisa documental dos *softwares* e do sistema operacional envolvido, assim como dos padrões e boas práticas de segurança da informação a serem respeitados. Deve ser atingida a maior compatibilidade possível, para que a solução obtida nesta pesquisa possa ser utilizada para o maior número possível de aplicações e clientes.

Para o acesso a *Internet*, essencial para as instalações, atualizações e exportação dos *backups*, estas máquinas instaladas para o sistema receberão compartilhamento do acesso da máquina que hospeda os servidores virtuais. Para os testes do sistema, deverá ser instalada ao menos uma estação de trabalho, igualmente virtual, acessível pelo sistema hospedeiro.

Será construído o ambiente das máquinas que farão parte da solução, sejam servidores ou clientes, de forma virtual utilizando *VirtualBox* instalado em máquina de propriedade do autor. Neste ambiente, serão criadas as máquinas, instalados nestas o sistema operacional e os *softwares* do projeto, criadas as redes virtuais, sendo estas configurações registradas na forma de roteiro de atividades.

Os algoritmos de verificação, informação ao administrador e *scripts*

resultantes serão especificados e criados, registrando-se. Da mesma forma para os algoritmos de teste e exercício dos sistemas, para estabelecimento das métricas de desempenho, estas conforme práticas comuns a sistemas *Linux*, a serem pesquisadas e igualmente documentadas. Mesmo procedimento será adotado para a metodologia de *backup*.

Estrutura do trabalho

Neste trabalho, é realizada uma revisão das tecnologias utilizadas no capítulo 1. A pesquisa das necessidades da solução e os *softwares* que atendem a estas necessidades é realizada no capítulo 2. O procedimento de construção do ambiente de teste, assim como a sua configuração, criação da máquina virtual, instalação de *softwares* e clonagem da máquina virtual estão no capítulo 3. As configurações específicas deste trabalho para o *software OpenLDAP* estão no capítulo 4, assim como para o *ISC Bind* estão no 5, para o *ISC DHCP* estão no 6 e para o *UCARP* estão no 7. No capítulo 8 estão as políticas criadas para a solução de alta disponibilidade, e as políticas para o *Backup* estão no capítulo 9. No capítulo 10 estão descritos os testes os ensaios de desempenho realizados, finalizando o trabalho com as conclusões descritas em capítulo com esta designação. Estão ainda adicionados a este trabalho 5 apêndices, sendo que o A contém procedimentos da criação de máquinas virtuais utilizando o *software VirtualBox*. No apêndice B está descrita a instalação do sistema operacional *ArchLinux* em máquina virtual. No apêndice C estão descritos os *scripts* criados para a automação e inteligência do sistema de alta disponibilidade. No apêndice D está descrita a criação e uso de uma autoridade certificadora local, e no apêndice final E está descrita a criação dos *scripts* que realizam o *Backup*.

Exclusões

- Não é objetivo deste trabalho estudo sobre tecnologias de virtualização de servidores, criar comparativos de desempenho entre tais tecnologias ou criar metodologias de escolha destes ambientes com relação ao seu desempenho,

fornecedor ou sistema de licenciamento;

- Não serão objetos de estudo os princípios de administração de sistemas operacionais, entre eles o *Linux* usado neste trabalho. Para referências podem ser consultados diversos *link*s disponíveis sobre o assunto, ou acessado o trabalho de graduação do autor disponível em https://drive.google.com/file/d/0B9YwgMsiyRCvVkVGZDBBZTY5U3M/view?usp=sharing;
- Fica restrito o desenvolvimento da solução proposta a ambientes confinados a redes fechadas, não sendo objetivo do projeto servir como *firewall* de rede ou de aplicação, mas fornecendo informação a clientes sobre estes recursos já presentes nestas redes. Não se limita desenvolvimentos futuros neste sentido;
- Não é objetivo deste trabalho o estudo sobre gerência de Certificados Digitais, limitando as instruções fornecidas àquelas necessárias à geração de dados indispensáveis aos propósitos deste estudo. Diversas referências podem ser consultadas a respeito, inclusive o site do projeto *OpenSSL*, da ICP-Brasil e o artigo do autor disponível em <https://drive.google.com/file/d/0B9YwgMsiyRCvRHA4VzBtXzYzMnc/view?usp=sharing>.
- Não é objetivo deste trabalho aprofundamento no estudo dos protocolos envolvidos, esquemas *LDAP*[2], gerenciamento de redes ou endereços;
- Os ensaios de desempenho não terão objetivo de realizar afinamentos na configuração para otimizá-lo, tampouco realizar comparativos com outras implementações, isto seria objeto de estudo à parte, considerando a implementação virtual minimalista realizada neste trabalho.

2 - Sigla do inglês *Lightweight Directory Access Protocol* (*LDAP*), como descrito na RFC 4511 (IETF, 2006)

1 REVISÃO BIBLIOGRÁFICA

Conforme Weber (2003, p.5), falhas são inevitáveis. Mas suas consequências podem ser evitadas por técnicas viáveis e de fácil compreensão. Ainda, técnicas tolerantes a falhas tem um alto custo, pela necessidade de adquirir e manter recursos para redundância ou alta disponibilidade. Porém, o custo/benefício de uma solução tolerante a falhas deve ser computada perante a imagem pública, impactos sociais e financeiros a uma empresa que fornece uma determinada solução de tecnologia, à qual esteja associada um contrato de fornecimento do qual possam depender não só bem-estar, diversão ou até vidas humanas.

É importante a diferença entre confiabilidade e disponibilidade. Ainda segundo Weber (2003, p.11):

> ... confiabilidade é a capacidade de atender a especificação, dentro de condições definidas, durante certo período de funcionamento e condicionado a estar operacional no início do período... Assim como a confiabilidade, a disponibilidade é uma medida de probabilidade. Disponibilidade é a probabilidade do sistema estar operacional num instante de tempo determinado.

Para este estudo, serão usados diversas metodologias, *softwares* e protocolos de rede. Os pontos críticos de sucesso para alcançar os objetivos propostos dependem da correta compreensão e interação destas tecnologias.

O sistema proposto depende de uma fonte confiável de informação. A partir desta, serão fornecidos autenticação, resolução de nomes, endereços e rotas e rede. Estes serviços não podem sofrer interrupções, portanto técnicas de alta disponibilidade serão aplicadas e com a mesma importância, as informações devem ser fornecidas de forma confidencial, portanto técnicas de criptografia igualmente serão importantes. A fonte de informação deverá ser replicada, garantindo a disponibilidade e integridade destas informações, essenciais aos serviços oferecidos.

O projeto proposto pelo autor pode ser classificado como um sistema tolerante a falhas, com redundâncias para *hardware*, *software* e informação. A falha

é mascarada e confinada, atuando a detecção de falhas nos *softwares* envolvidos pelos *scripts* de avaliação de alta disponibilidade, e o mesmo ainda é usado para detecção de falhas em *hardware*, pois seu mecanismo de eleição e o algoritmo a ser criado avaliarão constantemente a disponibilidade do sistema como um todo. Este mecanismo ainda deve informar ao administrador por *e-mail* as ocorrências em caso de falhas, reportando periodicamente o estado do sistema.

A principal conquista com um design como o proposto é que haverá, com a maior disponibilidade possível, um servidor disponível para fornecer um endereço de rede, rotas, autenticação de usuários e serviços de resolução de nomes a um cliente da rede que as solicite.

Assim exposto, serão revistas as tecnologias de serviços de diretório e sua replicação, criptografia em conexões de rede, resolução de nomes em endereços de rede, fornecimento de configurações de rede a clientes, técnicas de alta disponibilidade de serviços, a distribuição *Linux* de escolha e técnicas de controle de recursos.

1.1 *LDAP* e *OpenLDAP*

O *Lightweight Directory Access Protocol* (*LDAP*), como descrito na RFC 4511 (IETF, 2006), provê acesso a serviços de diretório distribuído, de acordo com o modelo de serviço e dados X.500. Elementos *LDAP* são baseados e construídos como os descritos para o protocolo de acesso a diretórios X.500 (DAP). A versão atual do protocolo é a 3, tendo a versão 2 sido considerada histórica pela RFC 3494 em março de 2003, mas ainda assim muito utilizada, como neste trabalho. O sincronismo das informações dos servidores *LDAP* fica garantida pela replicação usando um modelo *multi Master*, onde todos os servidores do nó *LDAP* são considerados primários.

Com base na tecnologia *LDAP*, foi desenvolvido o *software* livre *OpenLDAP*, que é o resultado de um esforço de colaboração visando obter uma suíte de aplicativos e ferramentas de desenvolvimento robusta, de qualidade

comercial, completa e sob a forma de *software* livre. O projeto é gerenciado por uma comunidade mundial de voluntários que usa a *Internet* para se comunicar, planejar e desenvolver a suíte *OpenLDAP* e a documentação relacionada. O projeto é uma atividade organizada da Fundação *OpenLDAP* (*OpenLDAP Foundation*, 2015).

1.2 SSL, TLS e *OpenSSL*

O protocolo *Secure Sockets Layer*(SSL) foi originalmente desenvolvido pela Netscape em 1994 para permitir transações de comércio seguras pela *Internet*, que requeriam encriptação para proteger os dados de seus consumidores, assim como garantir uma transação segura através da autenticação e integridade. Para garantir isso, o protocolo foi desenvolvido para funcionar sobre a camada TCP[3], possibilitando que protocolos de camadas superiores, como HTTP[4], POP[5], SMTP[6] e vários outros, pudessem operar sem alterações. Quando corretamente implementado, um terceiro observador pode apenas inferir a respeito dos pontos finais de comunicação, tipo de encriptação e volume de dados transferido, mas não poderá ver ou modificar seu conteúdo.

A primeira revisão pública disponível do protocolo foi a 2.0, seguida rapidamente pela 3.0, devido a diversas falhas descobertas. O protocolo SSL era proprietário da *Netscape*, assim a IETF (*Internet Engineering Task Force*), formou um grupo de trabalho como o objetivo de padronizar o protocolo, que resultou na publicação em janeiro de 1999 do *Transport Layer Security* (TLS) versão 1.0. Duas novas versões se seguiram, 1.1 em 2006 e a atual 1.2 em 2008. As versões SSL 3.0 e TLS 1.2 são muito similares, sendo suportadas pela maioria dos navegadores de *Internet*.

Conforme Ristić(2014), a família SSL/TLS tem cinco protocolos, SSL v2, SSL v3, TLS v1.0, TLS v1.1, e TLS v1.2. Ele recomenda que sejam considerados obsoletas as versões SSL v2 e SSL v3, quando em uso com HTTP, devido a

3 - TCP – *Transmission Control Protocol*, definido pela RFC 793
4 - HTTP - *Hypertext Transfer Protocol*, definido pelas RFC's 1945, 2068 e outras versões
5 - POP - *Post Office Protocol*, V3, definido pela RFC 1939
6 - SMTP - *Simple Mail Transfer Protocol*, definido pela RFC 5321

diversas vulnerabilidades conhecidas, assim como algumas cifras obsoletas e outras configurações específicas aos protocolos. Tais vulnerabilidades podem ser acompanhadas em <https://www.ssllabs.com/> e <https://www.OpenSSL.org>. Ainda, conforme a RFC 7568, a IETF (2015) recomenda que o SSLv3 não seja usado e considerado obsoleto pelas inúmeras vulnerabilidades.

Conforme a RFC 5246 (IETF, 2008), o principal objetivo do protocolo TLS é prover privacidade e integridade de dados para a comunicação entre duas aplicações. O protocolo é composto por duas camadas, TLS *Record Protocol* e TLS *Handshake Protocol*. TLS *Record Protocol*, que provê a segurança da conexão, está acima de algum protocolo de transporte, como o TCP, e encapsula outros protocolos, como HTTP entre outros, para os quais a presença do TLS é transparente.

O TLS *Handshake Protocol,* também encapsulado como os outros protocolos pelo TLS *Record Protocol*, permite que o cliente e o servidor se autentiquem, negociem um algoritmo de encriptação e chaves criptográficas antes que o tráfego de dados das aplicações se inicie. Uma vez estabelecida a versão e negociada a cifra, e sendo aceitos os certificados por ambas as partes conforme o caso, o cliente inicia a troca de chaves de criptografia por métodos assimétricos de chaves públicas, como pelos métodos RSA[7] ou *Diffie-Hellman*[8].

Conforme a RFC 5246 (2008), os parâmetros de criptografia são produzidos pelo TLS *Handshake Protocol*, num processo que envolve os seguintes passos:

- troca de mensagens do tipo *hello* (*ServerHello* e *ClientHello*) para mútua aceitação de algoritmos, troca de valores aleatórios e verificação de reinício de sessão;
- troca dos parâmetros criptográficos necessários que permitam ao cliente e

7 - Sistema assimétrico de criptografia baseado na dificuldade computacional de realizar a fatoração do produto de dois grandes números primos, implementado por Ron Rivest, Adi Shamir, e Leonard Adleman em 1977

8 - Método seguro de troca de chaves criptográficas sobre um canal público publicado por Whitfield Diffie e Martin Hellman em 1976.

servidor concordarem com a *premaster secret* gerada;

- troca de certificados e parâmetros criptográficos que permitam ao cliente e servidor autenticarem-se mutuamente;
- geração de *master secret* derivada da *premaster secret* e troca mútua de valores aleatórios;
- estabelecimento de parâmetros de segurança para a TLS *Record Layer*;
- verificação pelo cliente e o servidor se os mesmos valores de segurança foram calculados pelo par de comunicação, atestando que o *handshake* ocorreu sem que fosse adulterado por um atacante.

Uma vez que foi finalizado o *handshake*, e outra suíte de criptografia que não TLS_NULL_WITH_NULL_NULL foi escolhida, o tráfego de dados da aplicação pode iniciar.

O método RSA de troca de chaves tem sido dominante nas maiorias das implementações de TLS. Conforme a RFC 5246 (2008), o cliente gera uma pré-chave de 48 bits, encripta esta com a chave pública do servidor, e a envia ao mesmo. Este por sua vez a decripta com sua chave privada, e esta pré-chave simétrica é usada pelo cliente e servidor para gerar as chaves simétricas de sessão. Este método tem uma falha grave. O mesmo par de chaves assimétricas é usado para identificar o servidor e encriptar a pré-chave simétrica enviada ao servidor. Se um atacante obtiver acesso ao servidor e escutar a troca de chaves, poderá ler toda a sessão, mesmo se ainda não tiver acesso a chave privada, bastando gravar a sessão e quando obtiver a chave poderá decriptar toda a troca de dados.

Inversamente, o método *Diffie-Hellman* permite que a pré-chave simétrica seja negociada sem que se transfira a mesma pelo canal. A chave privada do servidor é usada para encriptar o *handshake,* para a troca dos parâmetros criptográficos que permitem ao cliente e servidor negociarem a pré-chave, mas esta nunca trafega entre o cliente e o servidor e portanto não pode ser interceptada, mesmo que o atacante tenha acesso a chave privada do servidor. Melhor, *Diffie-Hellman* pode ser usado para a troca periódica das chaves simétricas, tornando-as efêmeras. Assim, com chaves novas a cada sessão, mesmo que as chaves,

assimétricas ou simétricas, sejam comprometidas, uma sessão gravada não poderia ser decriptada.

Esta combinação do método *Diffie-Hellman*, onde as chaves de sessão não derivam do par assimétrico e são efêmeras é o que se chama "*Perfect Forward Secrecy"*. Mas para que a segurança do *Diffie-Hellman* seja confiável, não devem ser usados conjuntos *Diffie-Hellman* menores que 2048 bits ou a sessão será suscetível a ataques *Logjam* (ADRIAN et al. , 2015).

A versão TLS v1.0 é largamente considerada segura, com cuidados especiais na configuração. Conforme Ristić (2014) TLS v1.1 e 1.2 não possuem problemas de segurança conhecidos. Ristić (2014) ainda recomenda que TLS v1.2 seja o protocolo padrão, apesar de ainda poderem existir implementações de clientes suportando apenas versões anteriores, e a necessidade de compatibilidade deve ser verificada. SSL/TLS é construído primeiramente sobre TCP, mas há implementações sobre UDP[9].

Porém, conforme o site <https://weakdh.org/>, um grupo de cientistas da computação composto por David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Beguelin, e Paul Zimmermann das universidades *Inria Nancy-Grand Est*, *Inria Paris-Rocquencourt*, *Microsoft Research*, *Johns Hopkins University*, *University of Michigan*, e *University of Pennsylvania*, descobriu vulnerabilidades na forma como o método *Diffie-Hellman* falha na troca de chaves. Conforme Adrian et al (2015), esta falha chamada de ataque *Logjam*, é derivada do ataque *FREAK* ao SSL e age permitindo que um atacante *man-in-the-middle* baixe a criptografia de conexões vulneráveis a uma grade de 512 bits, quando estão habilitadas cifras que podiam ser exportadas para fora dos Estados Unidos, método usado na década de 90 devido a restrições impostas por leis daquele país.

Para os efeitos deste trabalho, será considerada a implementação do

9 - *User Datagram Protocol*, conforme definido pela RFC 768

protocolo TLS v1.2 oferecida pela suíte de aplicativos *OpenSSL*. Conforme o *OpenSSL Project* (2014), o projeto é um esforço colaborativo para desenvolver um conjunto de ferramentas completa, robusta, de qualidade comercial e livre para os protocolos *Secure Sockets Layer* (SSL) v2 e v3, para *Transport Layer Security* (TLS), e ainda uma biblioteca criptográfica de força máxima de uso geral.

1.3 DHCP

O *Dynamic HOST Configuration Protocol* (DHCP) é descrito pela RFC (*Request for Comments*) 2131, publicado pela IETF (*Internet* Engineering Task Force) em março de 1997. Este padrão fornece métodos para fornecimento de informações de configuração a clientes de uma rede TCP/IP[10].

O protocolo prevê que dois ou mais servidores podem compartilhar as informações fornecidas a clientes, mantendo consistência para os endereços fornecidos a quais clientes.

Porém, diferentemente do descrito na RFC 3074 DHCP *load balancing* (IETF, 2001), esta aproximação não se refere a balanceamento de carga, mas a *failover*, e não há impedimento na RFC 2131 DHCP *protocol* (IETF, 1997), contra dois ou mais servidores DHCP na mesma rede. Mas esta mesma RFC indica potenciais problemas de segurança, a serem objeto de novo e separado estudo, em que um servidor DHCP intruso poderia fornecer parâmetros de configuração a clientes de modo a comprometer a segurança e disponibilidade de recursos. Isto porque o protocolo DHCP é construído sobre pacotes de rede UDP sobre IP, cuja insegurança é intrínseca a sua construção.

1.4 DNS

O *Domain Name System* (DNS) consiste num mecanismo para nomear recursos da mesma forma que estes são usados em *HOSTs*, redes, famílias de

10 - *Transmission Control Protocol over Internet Protocol,* conforme mecanismo explicado na RFC 1180

protocolos, *Internet*s e organizações administrativas (RFC 1035, 1987).

Do ponto de vista do usuário, nomes de domínio são úteis como argumentos para um agente local, chamado resolvedor, que responde com informações sobre o domínio.

Desta forma, um usuário pode requisitar do agente informações como endereço de rede ou *e-mail* do responsável de um determinado segmento de rede. Do ponto de vista do agente resolvedor, o banco de dados representativo do domínio pode estar separado em diferentes servidores de nomes, assim como replicado entre dois ou mais, redundantemente.

O BIND, da *Internet Systems Consortion, inc.*, que conforme Moore(2004) é usado por 70% de todos os servidores DNS instalados, é um *software* livre que implementa os padrões do protocolo de serviços de nomes de domínio para a *Internet*. É uma implementação de referência para estes protocolos, assim como aplicável para a produção em sistemas de alto volume e confiança (ISC, 2015). Assim como o BIND, a ISC é responsável pelo *software* ISC DHCP, um *software* livre que implementa o padrão DHCP, para servidores, clientes e *relay agents*, suportando tanto IPv4 quanto IPv6, aplicável a sistemas de alto volume e confiança.

1.5 Alta disponibilidade e Redundância

Conforme Weber (2003, p.16), um sistema deve ser construído usando técnicas de tolerância a falhas quando prevenção e remoção destas não são suficientes, quando é exigido deste sistema alta confiabilidade ou alta disponibilidade. Usando componentes adicionais e algoritmos especiais, é implementada redundância a este sistema.

As duas técnicas de tolerância a falhas mencionadas por Weber (2003, p.16), de mascaramento e detecção, localização e reconfiguração, serão usadas neste projeto. Para isso, serão criados algoritmos especiais implementados em *scripts* de monitoramento, usando linguagens de programação disponíveis na

distribuição *Linux* selecionada. Ainda, serão incluídos componentes especiais para redundância da disponibilidade dos serviços oferecidos, descritos a seguir.

A RFC 2338 descreve o protocolo *Virtual Router Redundancy Protocol* (VRRP). Este protocolo define um método para eliminar um único ponto de falha inerente a redes definidas com um único *gateway* estático. Neste, o protocolo define o controle de endereços IP's associados a um roteador virtual, chamado de *Master*, que repassa os pacotes enviados a estes endereços. Este roteador virtual é o detentor de endereços, de responsabilidade resultante de um protocolo de eleição que assinala os IP's a um dos roteadores VRRP da rede. Em *Linux*, uma implementação deste protocolo é realizada pelo pacote VRRPd.

Conforme Raspet (2004), *Common Address Redundancy Protocol* (*CARP*) é uma versão melhorada do protocolo VRRP, para prover alta disponibilidade e redundância de rede. *CARP* foi desenvolvido por causa da possível existência de conflitos de propriedade intelectual entre o protocolo VRRP e o *Hot Standby Router Protocol* (HSRP) , de propriedade da *Cisco Systems*. *CARP* foi desenvolvido para ser fundamentalmente diferente, e uma das principais diferenças está no uso de criptografia, tendo sendo incluído na árvore de código *OpenBSD* e liberada a primeira versão disponível de *OpenBSD* com *CARP* em maio de 2004. Conforme Denis (2015), *UCARP* é a implementação portátil em espaço de usuário de *CARP* para *Linux*, e seus pontos chave são o baixo *overhead*[11], mensagens de sinalização criptografadas, interoperabilidade entre diferentes sistemas operacionais e não necessidade de conexões extras dedicadas entre os *HOSTs* físicos redundantes.

Keepalived é um *software* de roteamento escrito em C, uma das implementações do protocolo VRRP. A grande vantagem do projeto é prover um conjunto de facilidades simples e robustas para balanceamento de carga e alta disponibilidade para o sistema operacional *Linux* e infraestruturas baseadas em *Linux* (CASSEM, 2015).

11 - Pode ser considerado como qualquer combinação de excesso de tempo indireto de processamento, uso de memória, largura de banda ou qualquer outro excesso de recurso utilizado para se atingir um objetivo computacional

1.6 Distribuição *ArchLinux*

Conforme disponibilizado em sua página *web* oficial, o *ArchLinux* (2015):

> O Arch *Linux* é uma distribuição GNU/*Linux* i686/x86_64 de uso geral desenvolvida independentemente e versátil o suficiente para cumprir qualquer papel. O desenvolvimento é focado na simplicidade, minimalismo e elegância de código. O Arch é instalado como um sistema de base mínimo a partir do qual o usuário configura seu ambiente ideal, instalando apenas o que for requerido ou desejado para seus propósitos pessoais. Utilitários GUI (*Graphical User Interface*) de configuração não são providos oficialmente, e a maioria da configuração do sistema é realizada no terminal, editando simples arquivos de texto. Baseado no modelo de *rolling-release*, o Arch se esforça para se manter *bleeding edge*, e tipicamente oferece as últimas versões estáveis da maioria dos *softwares*.

A distribuição *Linux* escolhida permite que o sistema fique com apenas aqueles pacotes de *software* realmente necessários, tornando o sistema instalado pequeno e de rápida atualização. Sua arquitetura e filosofia de alterar com *patches* os *softwares* originais apenas em caso de extrema necessidade torna a instalação destes o mais próxima possível da revisão liberada como estável pelo responsável pelo *software*.

Este fato ganha importância em caso de necessidade de auditar o código, pois comparar os binários instalados por pacotes com os obtidos pela compilação a partir do código original é uma tarefa plenamente realizável. Ainda, esta distribuição permite que se construa os pacotes localmente, através de um sistema de compilação nomeado *Arch Build System*, ou *abs*, em que os *scripts* usados pela distribuição são disponibilizados integralmente, podendo ser visualizados e adaptados pelos usuários.

Neste sistema, através de ferramentas da distribuição, as versões liberadas pelos desenvolvedores são copiadas a partir do local de onde forem disponibilizadas, realiza-se a compilação conforme padrões nos arquivos de configuração, e o pacote construído é então instalado.

2 REQUISITOS E *SOFTWARES* DA SOLUÇÃO

O sistema proposto deve fornecer a partir de uma fonte única e consistente de dados:

- meios ágeis e automatizados de replicação, sincronismo e *backup* destes dados;
- endereços de rede e rotas;
- resolução de nomes;
- autenticação.

A pesquisa de um sistema de retaguarda que atenda a todos estes requisitos leva a serviços de diretório. Os serviços de diretório distribuído providos pelo protocolo *LDAP* atendem a estes requisitos, havendo diversas implementações disponíveis no ambiente *Linux*. São requisitos da implementação a ser escolhida:

- estar licenciada como *software* livre;
- deve permitir instalação no sistema operacional *Linux*;
- estar disponível para *download*, mesmo que apenas em código fonte;
- permitir replicação *MULTI-MASTER*;
- permitir conexão por TLS v1.2 com cifras consideradas seguras;
- permitir conexões de vários endereços de rede, por vários clientes ao mesmo tempo;
- permitir reconfiguração dinâmica;
- permitir *backup* e atualizações de dados sem interrupções no serviço.

Configurações utilizando bancos de dados SQL não proveriam meios de armazenamento ou compatibilidade nos requisitos de autenticação ou armazenamento de dados de usuários, o que extenderia a solução proposta com a aplicações de mais *softwares*, impactando a simplicidade, manutenção e robustez do sistema.

Foram encontradas as seguintes implementações do padrão *LDAP*, que atendem os requisitos, instaláveis no sistema *Linux*:

Quadro 1 - Algumas das implementações do protocolo *LDAP*.

Nome	Licença	Site do projeto	Replicação	TLS
389 *Directory Server*	GPL com possibilidade de *link* a não-GPL	http://www.port389.org	*MULTI-MASTER*	Conexões v1 com encriptação de até 256bits
OpenLDAP	*OpenLDAP Public Licence*	http://www.*OpenLDAP*.org	*MULTI-MASTER*	Funcionalidade TLS fornecida pelas suites *OpenSSL*, *gnutls* ou *moznss*
Apache *Directory server*	Apache *Licence* 2.0	http://directory.apache.org	*MULTI-MASTER*	Configuração de TLS dependente da versão de java em está sendo executado o servidor.
OpenDJ	CDDL 1.0	http://opendj.forgerock.org/	*MULTI-MASTER*	Configuração de TLS dependente da versão de java em está sendo executado o servidor, mas personalizável no servidor.
OpenDS	CDDL	https://java.net/projects/opends	*MULTI-MASTER*	Configuração de TLS dependente da versão de java em está sendo executado o servidor, mas personalizável no servidor.

Fonte: Coluna Site do projeto.

A implementação escolhida é o *OpenLDAP*, pelo fato de ser codificado em linguagem de programação C, portanto ser executado com maior rapidez e não depender de configuração e execução de máquina virtual java como o *OpenDJ* ou *OpenDS*, e permitir maior flexibilidade de configuração, ainda se utilizando de bibliotecas nativas do sistema, como *OpenSSL* e outras, não sofrendo as limitações de encriptação do 389DS.

O serviço DHCP oferece o fornecimento de parâmetros de configuração de rede aos clientes, tem como fonte de informação única e consistente a base de dados replicada do *LDAP*, atendendo aos requisitos. Ainda, o *software* a ser selecionado deve seguir a mesma metodologia de escolha do *LDAP*, conforme os requisitos a seguir:

- estar licenciada como *software* livre;
- deve permitir instalação no sistema operacional *Linux*;
- ter como base de dados o serviço *LDAP*;
- estar disponível para *download*, mesmo que apenas em código fonte;
- permitir compartilhamento dos endereços fornecidos a outros servidores DHCP (*failover*);
- permitir reconfiguração dinâmica;
- permitir *backup* e atualizações de dados sem interrupções no serviço.

Até setembro de 2015, momento da escrita deste trabalho, apenas o servidor DHCP da *Internet* Systems *Consortium* disponível em <https://www.isc.org/downloads/dhcp/> e o *opendhcp server* desenvolvido por Achal Dhir e disponível em <http://sourceforge.net/projects/dhcpserver/> atendem estes requisitos, sendo considerado para este projeto o servidor da ISC, por estar disponível virtualmente para todas as distribuições *Linux* hoje existentes, pelo sistema nativo de pacotes destas.

O serviço DNS oferece a tradução de nomes para endereços. O *software* a ser selecionado deve seguir a mesma metodologia de escolha do *LDAP*, conforme

os requisitos a seguir:

- estar licenciada como *software* livre;
- deve permitir instalação no sistema operacional *Linux*;
- deve suportar DNSSEC[12];
- ter como base de dados o serviço *LDAP*;
- estar disponível para *download*, mesmo que apenas em código fonte;
- permitir reconfiguração dinamica;
- permitir *backup* e atualizações de dados sem interrupções no serviço.

Estão disponíveis conforme a tabela a seguir os *softwares* que atendem aos requisitos:

Quadro 2 - Implementações de servidores DNS.

Nome	**Site do projeto**	**licença**	***Backend LDAP***	**DNSSEC**
LDAPdns	http://LDAPdns.sourceforge.net	GPL v2	desatualizado	não
bind	http://www.isc.org/software/bind	ISC	Sim, dlz e sdb	sim
powerdns	http://www.powerdns.com	GPL v2	desatualizado	sim

Fonte: Coluna Site do projeto.

O *software* selecionado é o BIND da ISC. Este *software* está disponível para a distribuição *Linux* escolhida, possui frequentes atualizações e atende aos requisitos.

Para que os clientes acessem os serviços a partir de uma fonte única, um único endereço de rede associado a estes serviços deve estar disponível. Para isso, em razão da necessidade de alta disponibilidade, foi escolhido o método do *software UCARP*, pelos motivos expostos no capítulo 3.

12 - *Domain Name System Security Extensions*, adiciona dados de autenticação da origem e integridade de dados ao sistema de nomes de domínio, conforme definido pela RFC 4033, disponível em <http://tools.ietf.org/html/rfc4033>. Acesso em 22 set. 2015.

3 CONSTRUÇÃO DO AMBIENTE DE TESTE

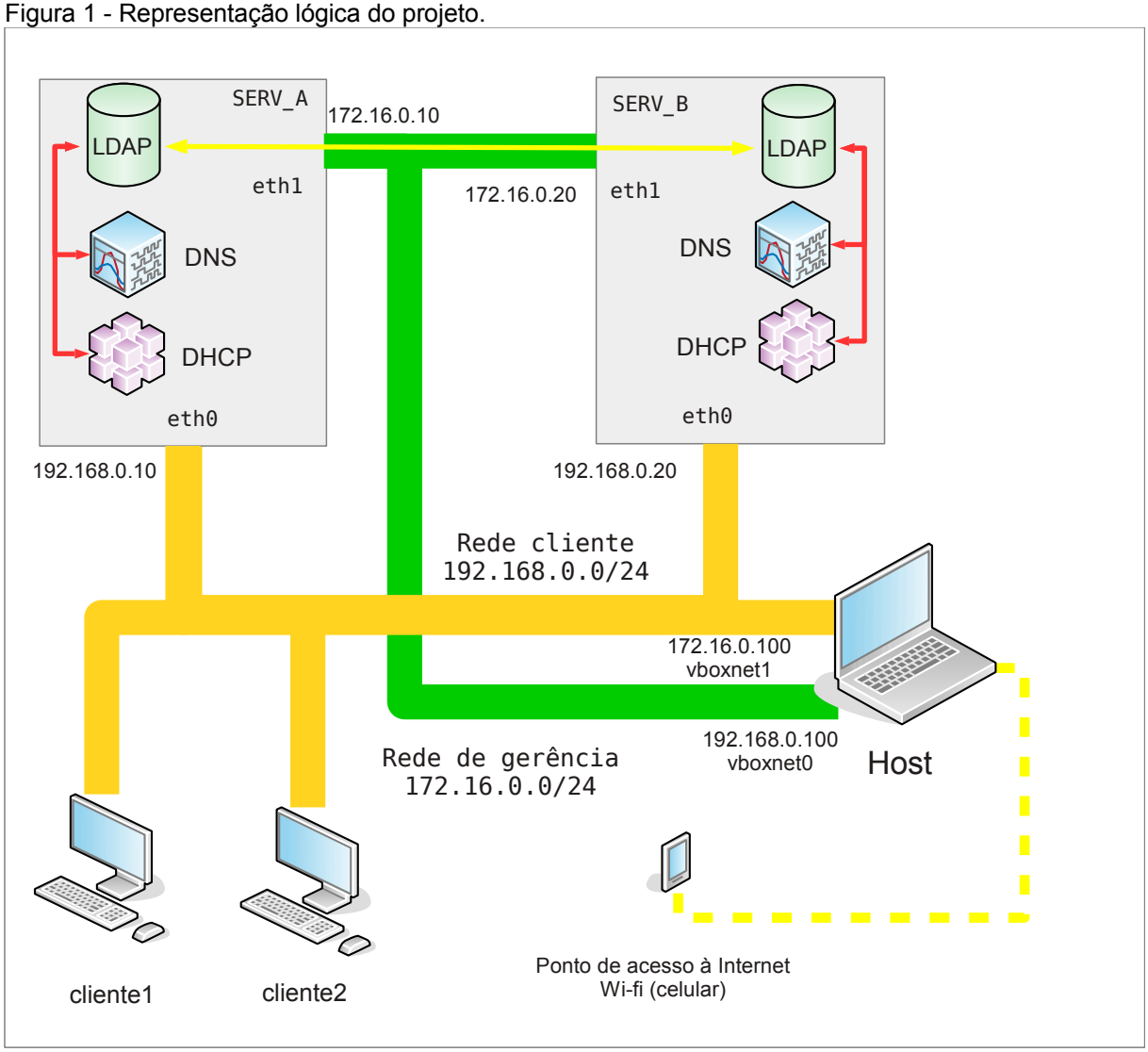
Para a construção do ambiente em que a solução será montada, devido questões de agilidade e custos, foi escolhido ambiente de virtualização *VirtualBox*. Este está disponível para a distribuição *ArchLinux*, possui constantes atualizações e está licenciado como *software* livre. Possui recursos que permitem a construção de ambientes de rede isolados, essenciais aos testes de tempo de resposta e uso de recursos necessários a este projeto, assim como flexibilidade no manejo dos recursos. Os servidores construídos para o projeto podem ser exportados para outros ambientes de virtualização, e reutilizados com algumas alterações em configuração.

O equipamento hospedeiro (*HOST*) é um notebook, da marca SAMSUNG modelo NP500P4C-AD1BR, composto por um processador *Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz*, com dois núcleos, 8GB de memória a 1333MHz, com disco SATA de 500GB. Está instalada a distribuição *ArchLinux*, 64 bits, mantida atualizada, com interface gráfica KDE. A instalação do *ArchLinux* no *HOST,* assim como os pacotes de *software* necessários ao correto funcionamento da interface gráfica fogem ao propósito deste trabalho.

3.1 Ambiente de teste

A representação lógica da rede do projeto está na figura 1. Conforme o diagrama, haverá uma rede independente conectando os servidores, na qual trafegarão os dados de gerência das aplicações, e outra rede por onde poderão se conectar os diversos clientes. As redes são separadas por questões de segurança. É boa prática de segurança da informação controlar os acessos a serviços críticos e isolar dados sensíveis, no caso, portas de administração, acesso privilegiado e dados de controle e atualização dos serviços.

Figura 1 - Representação lógica do projeto.



Fonte: Autor.

As redes apresentadas na figura 1 são virtuais, assim como as máquinas clientes e os servidores. Todos no ambiente *VirtualBox* executado no *HOST*.

3.2 Instalação do *VirtualBox*

Conforme a seção sobre *VirtualBox* do *Wiki* do *ArchLinux*(2015), a instalação dos pacotes necessários com o licenciamento GPL pode ser realizada pelo gerenciador de pacotes.

Mais detalhes sobre a instalação de pacotes no *ArchLinux* serão comentados durante a instalação dos servidores. Por agora, o comando a seguir serve ao propósito, executado a partir de um console no *HOST*:

```
#  pacman -Sy VirtualBox net-tools VirtualBox-guest-iso
```

Isto instalará o pacote e as dependências necessárias, como o pacote com os módulos do *kernel*, responsáveis pelo mecanismo de virtualização. O *VirtualBox* não utiliza componentes padrão do *kernel* do *Linux* para a virtualização, como o fazem outros *hypervisores*, entre eles o *Xen, KVM* ou *QEMU*. Porém, para que os módulos do *VirtualBox* possam funcionar, as extensões de virtualização do processador do *HOST* devem estar habilitadas. O módulo obrigatório para que máquinas virtuais possam ser executadas é o *vboxdrv*, porém alguns módulos e pacotes de *software* adicionais são necessários às tarefas de rede que farão parte deste projeto. Os comandos a seguir, executados com usuário privilegiado (root) configuram o arquivo que informa ao sistema quais módulos devem ser carregados, e o comando seguinte os carrega para que se possa iniciar o *VirtualBox*:

```
# echo -ne “vboxdrv \nvboxpci \nvboxnetadp \nvboxnetflt \n” > /etc/modules-load.d/VirtualBox.conf
# modprobe vboxdrv vboxpci vboxnetadp vboxnetflt
```

O usuário corrente não privilegiado, o $USER, que é referência à variável de sistema que contém a informação do *login* do usuário, precisa ser adicionado ao grupo *vboxusers*, para que a interface USB do *HOST* possa ser utilizada pelas máquinas virtuais. Lembrando que por prática de segurança não se usa usuário privilegiado para tarefas correntes em qualquer sistema operacional. O comando a seguir adiciona o usuário corrente ao grupo necessário:

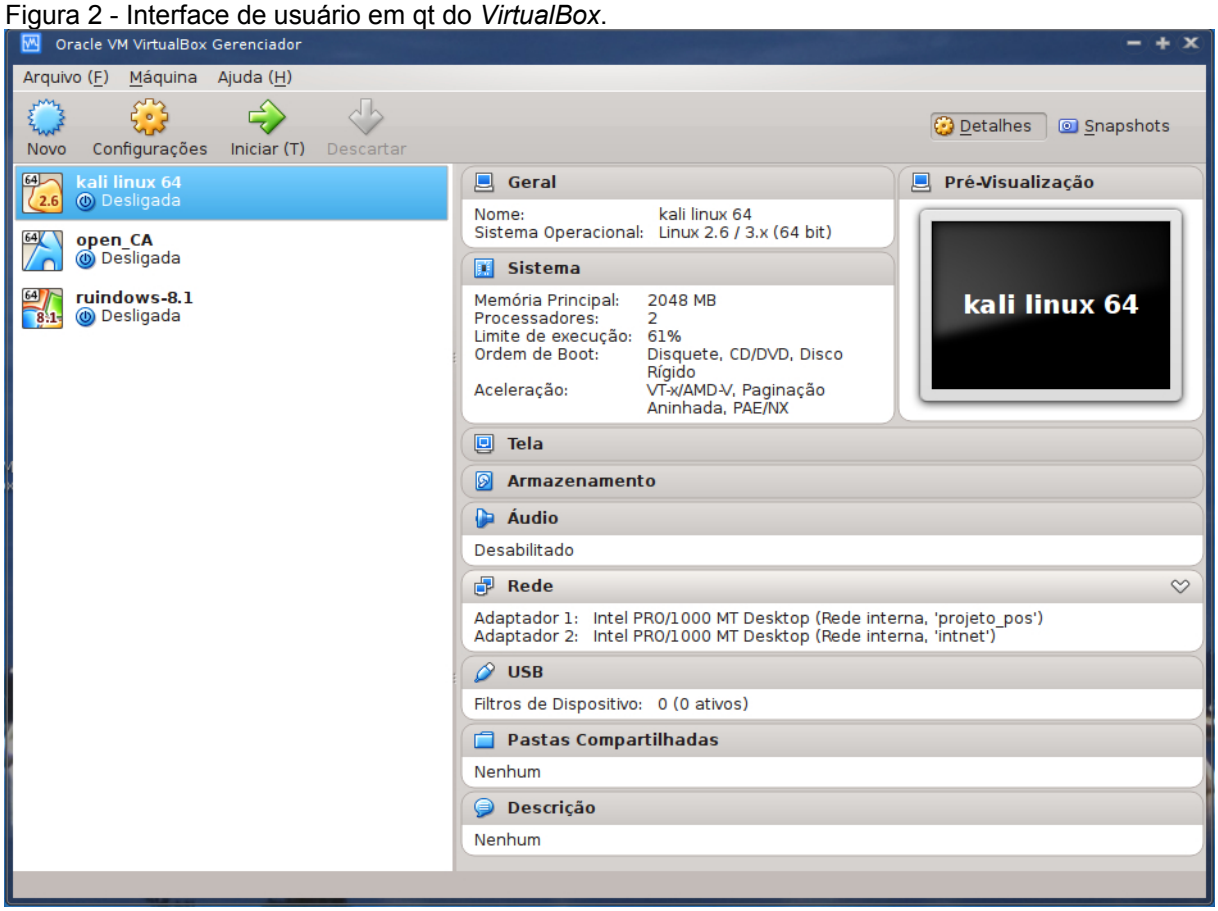
```
# gpasswd -a $USER vboxusers
```

Após um novo *login* no sistema do *HOST*, o *VirtualBox* já pode ser executado. Este *hypervisor* pode ser usado através de diversas interfaces, ou

iniciando as máquinas virtuais configuradas através de linha de comando.

Para o propósito deste trabalho, será usada a interface gráfica do usuário (*GUI*) baseada na biblioteca *qt*. Basta iniciar através do *link* criado em menu do sistema ou iniciá-lo com o comando *VirtualBox*, em qualquer console. A figura 2 a seguir mostra a interface inicial do VirtuaBox.

Figura 2 - Interface de usuário em qt do *VirtualBox*.



Fonte: Autor.

3.3 Configuração da infraestrutura

Para o projeto, serão utilizadas duas redes. Na *vboxnet0*, com endereço IPv4 de rede 192.168.0.0/24, está previsto o tráfego de clientes, por onde os mesmos poderão obter endereços por DHCP, resolver nomes pelo serviço DNS e utilizar os serviços *LDAP*, como armazenamento de informações e autenticação.

A rede *vboxnet1* é uma rede de gerência, por onde trafegarão os dados

de atualização do *LDAP*.

Para a criação destas redes no *VirtualBox*, acessa-se o menu Arquivo → Preferências → Rede. Seleciona-se a aba “Redes exclusivas de hospedeiro”. No quadro abaixo desta, clica-se no símbolo com o sinal + à direita no quadro, uma vez para cada rede. As redes criadas ficam listadas no quadro, como na figura a seguir.

Figura 3 - Cadastro das redes exclusivas de Hospedeiro.

Fonte: Autor.

Neste momento, a infraestrutura de rede virtual constante na figura 1 está criada. A rede de acesso à *Internet*, necessária para a instalação das máquinas virtuais, será provida por *NAT* através do *HOST*, que fará o papel de roteador da rede, não sendo necessária outra configuração adicional no *VirtualBox* para isso.

Para habilitar o acesso da rede *vboxnet0* usando o *HOST* como roteador, é necessário habilitar o repasse de pacotes e o mascaramento de endereços no *kernel* do *HOST*. Os comandos a seguir em algum console com usuário privilegiado realizam a tarefa:

```
# echo 1 > /proc/sys/net/ipv4/ip_forward
# iptables -t nat -A POSTROUTING -o wlp1s0 -j MASQUERADE
```

Nos comandos acima estão previstas políticas permissivas para a filtragem de pacotes no *HOST*, ou seja, *ACCEPT* para as *chain's INPUT*, *FORWARD* e *OUTPUT*. Não é escopo deste trabalho tratamento de políticas de *firewall*, exceto aquelas específicas para os servidores e serviços alvo deste estudo. Para maiores explicações sobre configurações de rede na distribuição *ArchLinux*, consulte a seção manual de configuração de rede do *Wiki*(2015) . Para configurações de *firewall,* consulte a seção do *iptables*(2015).

Para que as máquinas virtuais possam acessar a *Internet*, e o *HOST* acessar as redes configuradas, um endereço de rede deve ser atribuído às respectivas interfaces criadas pelo *VirtualBox* no *HOST*. Os comandos a seguir, realizados pelo usuário privilegiado em console, executam o procedimento.

```
# ifconfig vboxnet0 192.168.0.100 netmask 255.255.255.0 up
# ifconfig vboxnet1 172.16.0.100 netmask 255.255.255.0 up
```

As rotas de acesso a estas redes pelas interfaces designadas são incluídas automaticamente pelo comando, e não será necessário atribuir ao *HOST* neste momento rota padrão, pois esta será assinalada pelo método de acesso à *Internet* escolhido para o *HOST*. Não será coberto por este trabalho esta configuração, assim como a persistências destas configurações, tarefa que pode ser realizada consultando-se o manual de configuração de rede no *Wiki* do *ArchLinux*(2015).

3.4 Criação da máquina virtual serv_a no *VirtualBox*

A criação da máquina virtual está descrita no Apêndice A deste trabalho.

3.5 Instalação do *ArchLinux* na máquina virtual serv_a

O sistema operacional *ArchLinux* será instalado seguindo os procedimentos do manual de instalação disponível no site da distribuição, ciente do fato de que esta distribuição *Linux* tem uma forma pouco comum de instalação. Não há atualmente interfaces, gráficas ou não, para auxílio, o que leva a necessidade de conhecimentos do sistema para levar a instalação a cabo. Não é preocupação deste trabalho o detalhamento destas informações, haja visto que a profundidade dos temas aqui abordados pressupõe tais conhecimentos. Entretanto, para orientação e verificação dos requisitos deste trabalho, a instalação realizada encontra-se detalhada no Apêndice B – Instalação do *ArchLinux*.

3.6 Instalação dos *softwares* do projeto

O *software* base do projeto é o *OpenLDAP*, que servirá de base de dados para o ISC DHCP *Server* e para o ISC BIND. Estes serão acessíveis pelos clientes através de um endereço comum, atribuído a um dos servidores do projeto através de eleição controlada pelo *software UCARP*.

O comando a seguir instala todos estes *softwares*, disponíveis na distribuição escolhida, e suas dependências:

```
# pacman -Sy OpenLDAP dhcp UCARP openssh net-tools tcpdump ssmtp expect
```

3.7 Clonagem da máquina serv_a para serv_b

Com a máquina serv_a completa, faltando apenas as configurações específicas de cada *software*, é conveniente cloná-la, o que facilita e agiliza o processo de construção e replicação da solução.

Este procedimento pode ser feito através da interface gráfica do *VirtualBox*, ou em console do usuário que iniciou o *VirtualBox*. Para fins didáticos,

será executado em console, através do utilitário *vboxmanage* no *HOST*, pois este é o procedimento executado pela interface gráfica na interação com o usuário. Exige-se que a *vm* serv_a esteja desligada. O comando a seguir clona a *vm* e a registra automaticamente, disponibilizando o novo servidor como serv_b.

```
# vboxmanage clonevm serv_a --name template_projeto_ceub --register
```

Criando-se um modelo, a partir do qual apenas se especialize a configuração, agiliza-se o processo de entrega da solução, e facilita-se a reconstrução de uma das máquinas em caso de falha, perda ou comprometimento.

Finalizado o procedimento, repete-se esta cópia mas com nome de destino da cópia para serv_b. As configurações da vm clonada devem ser ajustadas para refletir as definições como serv_b. Depois de iniciada a *vm*, os arquivos a serem alterados são o /etc/hostname e /etc/hosts, com a definição do nome de máquina e IP, os arquivos em /etc/systemd/network, com as novas definições de endereçamento IP. Não há problemas com os endereços físicos das interfaces, pois o processo de clonagem mostrado prevê a sua reinicialização ao não ser informado explicitamente que sejam mantidos tais endereços. Em caso de uso do método pela interface gráfica, esta opção deve ser explicitamente selecionada.

4 CONFIGURAÇÃO DO *OPENLDAP*

O *OpenLDAP* é a base de dados única escolhida para o projeto. Será descrita sua instalação na máquina serv_a, e ajustando-se os nomes relativos à máquina, estas configurações deverão ser aplicadas à máquina serv_b.

Após, será realizada a configuração da replicação, e depois da interface de administração, em cada máquina, que poderá ser acessada para alterar ou visualizar dados do *OpenLDAP*. Neste modo de operação, qualquer informação alterada num servidor será refletida no outro, como se fosse uma única base de dados.

4.1 Instalação

O *OpenLDAP* foi instalado em passos anteriores, mas diversos passos de configuração ainda são necessários, conforme a seção *OpenLDAP* do *Wiki* do *ArchLinux*(2015). Tais configurações iniciais estão localizadas no arquivo /etc/openldap/slapd.conf, que será compilado gerando o novo esquema de configuração *cn=config* em /etc/openldap/slapd.d/ , através do utilitário *slaptest*.

Para os efeitos deste projeto, será permitida conexão remota ao *OpenLDAP* apenas de forma criptografada, por TLSv1.2. Para conexões locais, será permitida conexão não criptografada, para reduzir o *overhead* de processamento.

As alterações necessárias, para a criação de um banco de dados inicial, são o “suffix” e o “rootdn”. Respectivamente, o “suffix” contém normalmente o domínio da localidade de instalação e o “rootdn” é o usuário raiz do servidor *LDAP*. Para este projeto, será considerado o domínio “kirdeika.org” e o usuário raiz “root”. As seções deste arquivo ficam assim definidas:

```
suffix          “dc=kirdeika,dc=org”
rootdn          “cn=root,dc=kirdeika,dc=org”
```

O terceiro parâmetro importante é a senha do usuário raiz. Para maior segurança da instalação, esta senha não deve ser deixada em texto plano no arquivo de configuração, e pode ser criada criptografada através do utilitário *slappasswd*. Este pede que seja digitada a senha a ser criptografada, e retorna um *hash* que pode ser usado como valor do parâmetro de senha. Os comandos a seguir exemplificam o uso:

```
root@serv_a ~ # slappasswd
New password:
Re-enter new password:
{SSHA}KpogLmuiypsxxb0DBYF7Ns2g4YBMYEXD
```

Então, o valor retornado pode compor o parâmetro no arquivo:

```
rootpw      {SSHA}KpogLmuiypsxxb0DBYF7Ns2g4YBMYEXD
```

Outra opção, é gerar a senha e alterar o arquivo, numa única operação, como a seguir.

```
# sed -i "s/rootpw.*/rootpw  $(slappasswd)/" /etc/openldap/slapd.conf
```

Após, deve-se adicionar alguns esquemas comuns à configuração. Adiciona-se as seguintes linhas ao arquivo /etc/openldap/slapd.conf :

```
include /etc/openldap/schema/core.schema        ### linha existente
include /etc/openldap/schema/cosine.schema      ###> adicionar as seguintes
include /etc/openldap/schema/inetorgperson.schema
include /etc/openldap/schema/nis.schema
```

Para a criação dos índices mais comuns, adiciona-se ao mesmo arquivo:

```
index   objectClass     eq
index   uid             pres,eq
index   mail            pres,sub,eq
index   cn              pres,sub,eq
index   sn              pres,sub,eq
index   dc              eq
```

Deve ser criado um arquivo básico de configuração do banco de dados,

os comandos a seguir realizam a tarefa, com a correta aplicação de dono do arquivo.

```
# cp /etc/openldap/DB_CONFIG.example /var/lib/openldap/openldap-
data/DB_CONFIG
# chown -R ldap:ldap /var/lib/openldap
# chown -R ldap:ldap /etc/openldap
```

Um banco de dados inicial deve ser criado. Porém, o “unit” de controle do *OpenLDAP* fornecido para o *ArchLinux* não contém a linha que finaliza a execução do *OpenLDAP*, e deve ser alterado. O arquivo final /usr/lib/systemd/system/slapd.service deve ficar como a seguir.

```
[Unit]
Description=OpenLDAP server daemon

[Service]
Type=forking
ExecStart=/usr/bin/slapd -u ldap -g ldap -h "ldap://127.0.0.1:389/
ldaps://:636/ ldapi:///"
ExecStop=/usr/bin/kill -INT $MAINPID

[Install]
WantedBy=multi-user.target
```

Para que as configurações criadas sejam devidamente convertidas ao novo formato *cn=config*, usado pelo *OpenLDAP* a partir da versão 2.4, é fornecida a ferramenta *slaptest*. O *OpenLDAP* deve ser inciado uma vez, para a criação dos arquivos de banco de dados e finalizado. As configurações são geradas a partir do comando a seguir, com o devido acerto de propriedades dos arquivos criados.

```
# systemctl daemon-reload
# systemctl start slapd
# systemctl stop slapd
# slaptest -f /etc/openldap/slapd.conf -f /etc/openldap/slapd.d/
# chown -r ldap:ldap /etc/openldap/slapd.d
```

A ferramenta *slaptest* não configura o acesso de usuários corretamente, portanto modificações manuais são necessárias. Com o servidor *LDAP* parado, edita-se o arquivo /etc/openldap/slapd.d/cn=config/olcDatabase={0}config.ldif, remove-se as duas primeiras linhas relativas ao hash de verificação cíclica de redundância (CRC) do arquivo, e adiciona-se as regras de acesso que permitirão

editar o *LDAP*. O arquivo fica como a seguir. Remover as duas linhas iniciais impede a verificação do arquivo por CRC, e portanto as alterações não serão negadas pelo *daemon slapd*.

```
 - deletar estas duas linhas seguintes
# AUTO-GENERATED FILE - DO NOT EDIT!! Use ldapmodify.
# CRC32 ...

 - deletar esta linha a seguir
olcAccess: {0}to *  by * none

# inserir este trecho
olcAccess: {0}to *
 by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth" write
 by dn.base="cn=root,dc=kirdeika,dc=org" write
 by self write
 by * read
```

A primeira linha do *olcAccess* permite que se conecte ao *LDAP* por *API*, sem necessidade ou disponibilidade de conexão de rede. Devem ser observados espaços finais nas linhas do *olcAccess*, especificamente nas que se iniciam com “by”. Deve ser constatada a presença de um espaço adicional ao fim da linha para correta interpretação pelo *OpenLDAP*.

Então, o serviço *LDAP* pode ser habilitado e iniciado.

```
# systemctl enable slapd
# systemctl start slapd
```

O banco de dados deve ser inicializado com a entrada raiz e o usuário root. Cria-se o arquivo entrada_inicial.dif com o conteúdo a seguir:

```
dn: dc=kirdeika,dc=org
objectClass: dcObject
objectClass: organization
dc: kirdeika
o: KirdeiKa
description: Diretório do projeto de Pós-graduação - CEUB

dn: cn=root,dc=kirdeika,dc=org
objectClass: organizationalRole
cn: root
description: Gerente do Diretório
```

A inserção destas informações pode ser realizada com o comando a seguir:

```
# ldapadd -x -D "cn=root,dc=kirdeika,dc=org" -W -f entrada_inicial.ldif
```

Será solicitada a senha configurada para o usuário raiz em passo anterior. Para a configuração de controle de acessos, deve ser criado o arquivo acerto_acessos.ldif, com o conteúdo a seguir.

```
dn: olcDatabase={0}config,cn=config
changetype: modify
delete: olcAccess
-
add: olcAccess
olcAccess: {0}to *
 by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth"
manage
 by dn.base="cn=root,dc=kirdeika,dc=org" manage
 by self write
 by * read
```

Então, estas regras de acesso podem ser aplicadas ao *OpenLDAP* da mesma forma, com o comando a seguir.

```
# ldapadd -x -D "cn=root,dc=kirdeika,dc=org" -W -f acerto_acessos.ldif
```

4.2 Configuração TLS

Para o projeto, será usada a mesma estrutura de chaves públicas que seria usada em caso de implantação real do sistema.

Será criada uma chave privada para o servidor, esta será usada para criar uma solicitação de assinatura de certificado (*certificate sign request,* CSR), nesta solicitação estarão contidos os nomes de domínio totalmente qualificados (FQDN) dos servidores envolvidos na replicação da base *LDAP*, para que o mesmo certificado possa ser usado nestes, facilitando a configuração e reconhecimento deste certificado pelos clientes. Esta CSR deverá ser assinada pela autoridade certificadora, e o certificado gerado será instalado nos servidores, conjuntamente

com o certificado da autoridade.

Para que o certificado assinado possa ser reconhecido e aceito pelos clientes, o certificado da autoridade deverá ser instalado em todos os clientes que acessarão a base *LDAP*. Isto também deve ser feitos no servidores, para o correto funcionamento da replicação.

Todo o trabalho de manipulação de certificados pode ser verificado no Apêndice D – Autoridade Certificadora Local.

Após a instalação do certificado, para o devido funcionamento da conexão TLS é exigido um arquivo com parâmetros para troca de chaves efêmeras *Diffie-Hellman*. O comando a seguir gera tal arquivo.

```
# openssl dhparam -dsaparam -out /etc/openldap/serv_a_dh.pem 4096
```

Então, as informações relativas ao TLS podem ser inseridas no *LDAP*, através da configuração dinâmica *cn=config*. Nesta, usa-se a ferramenta ldapadd ou ldapmodify, com as modificações contidas em arquivos LDIF[13]. O arquivo /etc/openldap/tls.ldif, cujo conteúdo está a seguir, mostra as alterações a serem realizadas.

```
dn: cn=config
changetype: modify
add:olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/openldap/chave_servidor.pem
-
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/openldap/serv_a.kirdeika.org.pem
-
add: olcTLSCACertificateFile
olcTLSCACertificateFile: /etc/openldap/kirdeika-tcc.pem
-
add: olcTLSDHParamFile
olcTLSDHParamFile: /etc/openldap/serv_a_dh.pem
-
add:olcTLSCipherSuite
olcTLSCipherSuite: HIGH:+TLSv1.2:-TLSv1.1:-TLSv1.0:-SSLv3:-SSLv2
-
```

13 - *LDAP Data Interchange Format,* conforme definido pela RFC 2849.

Neste arquivo, estão indicados os caminhos dos certificados da autoridade certificadora, o certificado assinado, da chave privada do servidor, do arquivo de parametros *Diffie-Hellman*, e conforme o exposto sobre vulnerabilidades do *OpenSSL*, são desabilitados os protocolos vulneráveis e habilitadas apenas as cifras consideradas seguras.

Conforme Gil (2012, p. 180), ainda neste arquivo, podem ser definidos os níveis de fator de força de encriptação, como no mínimo 256 bits para os *ciphers* para conexões via TLS e desabilitados para conexões locais, que não usam a rede.

Pode-se então inserir as alterações na base *LDAP* com a ferramenta *ldapmodify*. O comando a seguir executa a tarefa.

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f /etc/openldap/tls.ldif
```

As modificações implementadas, a escolha das cifras, do tipo de certificado e do tamanho da grade de encriptação foram realizadas conforme indicações de segurança da planilha de fraudes de TLS da OWASP(2015) e da página de manual do slapd.conf, parte da instalação do *OpenLDAP*. Porém conforme Dreyfus (2014), a maioria dos ataques possíveis para o TLS quando em conjunto com HTTP, não são possíveis com *LDAP* e diversos outros protocolos. Desta forma, não há necessidade de desabilitar a compressão das conexões, e ainda não seria possível pois não há esta configuração para *OpenLDAP*, até o momento da escrita deste trabalho.

4.3 Configuração de LOG

Nas configurações padrão do *ArchLinux*, o *log* gerado pelo sistema não é armazenado de forma definitiva. Para efeitos de auditoria e pesquisa de falhas posterior, isso não é aceitável.

O sistema de *log* padrão, o *systemd-journald*, recebe os *logs* de forma

unificada de todos os processos, mas permite pelas ferramentas fornecidas a visualização separada de cada *log*, por filtragem. Para facilitar a construção posterior dos *scripts* de monitoramento, o ideal é que os *logs* de cada aplicação sejam armazenados em arquivos distintos. Para que o *LOG* gerado pelo *OpenLDAP*, ou pelas outras aplicações, seja direcionado a arquivo separado, alterações no sistema de *journal* do *systemd* e a instalação do pacote *syslog-ng* devem ser realizadas. Por padrão, se o *log* do *OpenLDAP* for habilitado, este envia as mensagens para o *log* do sistema, com a marca local4 e o *systemd-journald* captura esta informação. Conforme o *Wiki* do *Syslog-ng ArchLinux*(2015), as mensagens de sistema desde a versão 216 do *systemd* não são mais direcionadas ao *syslog* por padrão. Assim, a opção *ForwardToSyslog=yes* deve ser adicionada ao arquivo de configuração de *journal* /etc/systemd/journald.conf, e o *unit syslog-ng.service* deve ser habilitado e iniciado. Deve ainda ser criado um diretório para manter a organização. Os comandos a seguir realizam estas alterações:

```
# pacman -Sy syslog-ng
# echo  ForwardToSyslog=yes >> /etc/systemd/journald.conf
# systemctl enable syslog-ng
# mkdir /var/log/slapd
```

Porém, para que as mensagens sejam filtradas e direcionadas ao arquivo único, o arquivo de configuração /etc/syslog-ng/syslog-ng.conf deve ter alterações. Na seção *options*, deve ser inserida uma linha com “flush_lines (0);” , e mais adiante, as seguintes linhas:

```
filter f_local4         {facility(local4); };
destination slapd { file("/var/log/slapd/slapd.log"); };
log {source(src); filter(f_local4); destination(slapd); };
```

Então, o *syslog-ng* pode ser iniciado e o *journald* deve ser reiniciado com os comandos a seguir.

```
# systemctl start syslog-ng
# systemctl restart systemd-journald
```

A diretiva do *cn=config olcLOGFile* não tem utilidade em *Linux* pois esta foi desenvolvida para uso em sistemas sem *syslog*, como *Windows*.

4.4 Configuração do cliente local, testes e debug

Para habilitar o acesso pelas ferramentas locais, o cliente local *LDAP* deve ser configurado. O arquivo /etc/openldap/ldap.conf deve ficar como a seguir.

```
BASE    dc=kirdeika,dc=org
URI    ldaps://serv_a.kirdeika.org:636/ ldap://127.0.0.1:389/

TLS_CACERT /etc/openldap/kirdeika-tcc.pem
```

Testes de conexão podem ser realizados pela ferramenta *ldapsearch*. A consulta a seguir retorna todos os dados cadastrados no *LDAP*, e a seguinte as configurações.

```
# ldapsearch -H ldaps:/// -D cn=root,dc=kirdeika,dc=org -W
# ldapsearch -H ldaps:/// -D cn=root,dc=kirdeika,dc=org -W -b cn=config
```

Em caso de problemas, iniciar o *OpenLDAP* em modo *debug* permite visualizar o erro com maior facilidade e objetividade, o que pode ser realizado com o comando a seguir.

```
# slapd -u ldap -g ldap -h “ldapi:/// ldap:/// ldaps:///” -d -1
```

4.5 Configuração do servidor serv_b

Os itens de configuração 7.1 a 7.4 constantes neste trabalho deverão ser aplicados ao servidor serv_b, devendo os arquivos de chave privada, certificados e arquivos de configuração LDIF serem reaproveitados, copiando-se de um servidor a outro no mesmo diretório, alterando-se apenas o nome dos arquivos de certificado para o correspondente servidor serv_b.

4.6 Configuração da replicação

É imperativo que a configuração da replicação seja realizada antes que qualquer dado seja inserido no banco. Isto porque o modo de replicação *N-WAY*

MULTI-MASTER escolhido faz com que cada alteração em um nó, ou seja, servidor *OpenLDAP* pertencente ao conjunto de servidores com a replicação configurada, seja propagada a todos os outros, e o controle sobre a validade deste dado incide a partir do parâmetro de tempo. Assim, a informação mais recente é replicada, e se houver alguma informação com atraso em algum nó, esta será excluída e substituída pela mais recente.

Da mesma forma, o sincronismo de tempo entre os servidores é fundamental. Para os efeitos deste projeto, o sincronismo realizado pelo *VirtualBox* será suficiente. Em ambientes reais um serviço NTP instalado em algum componente na rede, como um roteador ou servidor específico com redundância, inclusive nos mesmos nós que compõem a solução apresentada, pode ser implementado.

Igualmente, a resolução de nomes independente do serviço DNS oferecido deve ser configurada, de forma estática nos servidores da solução, utilizando-se dos arquivos /etc/hosts para tal. Inclui-se nestes as configurações a seguir.

```
172.16.0.10 serv_a.kirdeika.org serv_a
172.16.0.20 serv_b.kirdeika.org serv_b
```

Os IP's das interfaces de administração foram escolhidos justamente para que se isole o tráfego de sincronismo entre os servidores da rede de acesso de clientes. Esta abordagem não é imperativa, e foi escolhida para efeitos de métricas posteriores, permitindo isolar o tráfico de rede de replicação e administração. Ainda, esta replicação é realizada apenas em conexões TLSv1.2.

Para habilitar a replicação em modo *MULTI-MASTER*, conforme o manual do *OpenLDAP* seção *Replication*(2015), habilitando a replicação para configuração e dados, cria-se o arquivo /etc/openldap/replica_serv_a.ldif com o seguinte conteúdo:

```
dn: cn=config
```

```
changetype: modify
replace: olcServerID
olcServerID: 1 ldaps://serv_a.kirdeika.org:636/
olcServerID: 2 ldaps://serv_b.kirdeika.org:636/

dn: olcDatabase={0}config,cn=config
changetype: modify
add: olcRootPW
olcRootPW: {SSHA}i4+ZH5dkqhLbjtDbXpAhdLBzIzrdvz1q

dn: cn=module,cn=config
changetype: add
olcModuleLoad: syncprov
objectClass: olcModuleList
cn: module
olcModulePath: /usr/lib/openldap
olcModuleLoad: syncprov.so

dn: olcOverlay=syncprov,olcDatabase={0}config,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov

dn: olcDatabase={0}config,cn=config
changetype: modify
add: olcSyncRepl
olcSyncRepl: rid=001 provider=ldaps://serv_a.kirdeika.org:636/
 binddn="cn=config" bindmethod=simple credentials=$senha
 searchbase="cn=config" type=refreshAndPersist retry="5 5 300 5" timeout=1
olcSyncRepl: rid=002 provider=ldaps://serv_b.kirdeika.org:636/
 binddn="cn=config" bindmethod=simple credentials=$senha
 searchbase="cn=config" type=refreshAndPersist retry="5 5 300 5" timeout=1
-
add: olcMirrorMode
olcMirrorMode: TRUE

dn: olcOverlay=syncprov,olcDatabase={1}bdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov
dn: olcDatabase={1}bdb,cn=config
changetype: modify
add: olcLimits
olcLimits: dn.exact="cn=root,dc=kirdeika,dc=org" time.soft=unlimited
time.hard=unlimited size.soft=unlimited size.hard=unlimited
-
add: olcSyncRepl
olcSyncRepl: rid=004 provider=ldaps://serv_a.kirdeika.org:636/
 binddn="cn=root,dc=kirdeika,dc=org" bindmethod=simple
 credentials=$senha searchbase="dc=kirdeika,dc=org"
 type=refreshOnly interval=00:00:00:10 retry="5 5 300 5" timeout=1
olcSyncRepl: rid=005 provider=ldaps://serv_b.kirdeika.org:636/
 binddn="cn=root,dc=kirdeika,dc=org" bindmethod=simple
 credentials=$senha searchbase="dc=kirdeika,dc=org"
```

```
 type=refreshOnly interval=00:00:00:10 retry="5 5 300 5" timeout=1
-
add: olcMirrorMode
olcMirrorMode: TRUE
-
add: olcDbIndex
olcDbIndex: entryCSN,entryUUID eq
```

Deve ser substituído o campo $senha por uma senha forte, com ao menos 8 caracteres, contendo alfanuméricos e símbolos. Após, aplica-se a configuração deste arquivo no servidor serv_a, com o comando a seguir:

```
# ldapmodify -Y EXTERNAL -H ldapi:/// -f replica_serv_a.ldif
```

Para testes de replicação, aplica-se o arquivo /etc/openldap/adiciona_usuario.ldif a seguir no serv_a, e busca-se o mesmo no serv_b, com os comandos a seguir:

```
dn: cn=dhcp,dc=kirdeika,dc=org
changetype: add
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: dhcp
description: Usuario para o bind do DHCP
userPassword: {SSHA}xqfcjHFbO8piRYoQXbz0c0txRKFdAEAz
```

```
# ldapsearch -H ldaps://serv_a.kirdeika.org:636/ -D
"cn=root,dc=kirdeika,dc=org" -W -b "dc=kirdeika,dc=org" cn=dhcp
# ldapsearch -H ldaps://serv_b.kirdeika.org:636/ -D
"cn=root,dc=kirdeika,dc=org" -W -b "dc=kirdeika,dc=org" cn=dhcp
```

Deve-se obter o mesmo resultado de ambos os servidores, como a seguir:

```
# extended LDIF
#
# LDAPv3
# base <dc=kirdeika,dc=org> with scope subtree
# filter: cn=dhcp
# requesting: ALL
#

# dhcp, kirdeika.org
dn: cn=dhcp,dc=kirdeika,dc=org
objectClass: simpleSecurityObject
objectClass: organizationalRole
```

```
cn: dhcp
description: Usuario para o bind do DHCP
userPassword:: e1NTSEF9eHFmY2pIRmJPOHBpUllvUVhiejBjMHR4UktGZEFFQXo=

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

4.7 Interface de administração

A administração da base de dados e configurações do *OpenLDAP* pode ser realizada de várias formas, através de clientes remotos, com ou sem interfaces gráficas, ou através de ferramentas especializadas instaladas no próprio servidor, como *phpldapadmin*, disponível em <http://phpldapadmin.sourceforge.net/> ou *LDAP Account Manager*, disponível em <https://www.ldap-account-manager.org/lamcms/>.

Para o projeto, foi escolhido o *LDAP Account Manager*, por estar disponível para a distribuição *ArchLinux* pelo AUR, pela integração com as definições de DHCP e interface mais amigável, sendo a ferramenta acessada por navegador, de qualquer estação cujo endereço de origem pode ser configurado na ferramenta.

Conforme o manual do *Ldap-Account-Manager* (LAM, 2015), as dependências de *software* que devem ser instaladas incluem um servidor *Web*, que pode ser o Apache ou NGINX, interpretador da linguagem PHP com versão maior ou igual a 5.2.4, com módulos *LDAP, gettext, xml, OpenSSL* e opcionalmente *mcrypt*.

Também é requerido *Perl* se o módulo *lamdaemon* for usado. Obviamente, uma conexão com um servidor *LDAP* e um navegador *web* recente é imperativo. Para os efeitos deste estudo, serão usados os navegadores Firefox versão 39 e Google Chome versão 41.0.2272.89, instalados no *HOST.*

As dependências podem ser instaladas pelo gerenciador de pacotes do *ArchLinux*. O comando a seguir instala estas dependências:

```
# pacman --needed -Sy apache php php-mcrypt perl php-apache php-ldap
```

As dependências devem ser configuradas a começar pelo interpretador PHP. Conforme o manual do LAM, o arquivo /etc/php/php.ini deve ser alterado para habilitar certos módulos necessários ao LAM, assim como definir as permissões de acesso a determinados diretórios. As seções alteradas devem ficar como a seguir:

```
extension=ldap.so
extension=mcrypt.so
extension=openssl.so
extension=zip.so
open_basedir =
/srv/http/lam:/srv/http/lam/sess:/srv/http/lam/tmp:/var/log/lam:/tmp
```

O servidor *web Apache*, escolhido para o projeto, conforme o *Wiki* do *ArchLinux* na seção *Apache HTTP Server* (2015), após instalado deve ser configurado a começar pelo arquivo /etc/http/conf/httpd.conf, com as alterações a serem realizadas relacionadas a seguir, na sua forma final:

```
ServerAdmin      kirdeikajr@gmail.com
ServerName  serv_a.kirdeika.org:80
# Include conf/extra/httpd-userdir.conf
LoadModule ssl_module modules/mod_ssl.so
LoadModule socache_shmcb_module modules/mod_socache_shmcb.so
Include conf/extra/httpd-ssl.conf
# LoadModule mpm_event_module modules/mod_mpm_event.so
LoadModule mpm_prefork_module modules/mod_mpm_prefork.so
LoadModule dir_module modules/mod_dir.so
LoadModule php5_module modules/libphp5.so
Include conf/extra/php5_module.conf
```

No arquivo /etc/httpd/conf/extra/httpd-default.conf as alterações são as seguintes:

```
ServerTokens Prod
ServerSignature Off
UseCanonicalName On
```

A seguir, as configurações de TLS são realizadas no arquivo /etc/httpd/conf/extra/httpd-ssl.conf. Copia-se os arquivos de chave privada e

certificados para o diretório de configuração do Apache, alterando-se as permissões conforme os comandos a seguir:

```
# cp /etc/openldap/chave_servidor.pem /etc/openldap/kirdeika-tcc.pem \
/etc/openldap/serv_a.kirdeika.org.pem /etc/httpd/conf/
# chown http:http /etc/httpd/conf/*.pem
# chmod 400 /etc/httpd/conf/chave_servidor.pem
# chmod 444 /etc/httpd/conf/kirdeika-tcc.pem
# chmod 444 /etc/httpd/conf/serv_a.kirdeika.org.pem
```

As alterações no arquivo /etc/httpd/conf/extra/httpd-ssl.conf ficam como a seguir:

```
SSLCertificateFile "/etc/httpd/conf/serv_a.kirdeika.org.pem"
SSLCertificateKeyFile "/etc/httpd/conf/chave_servidor.pem"
SSLCipherSuite HIGH:+TLSv1.2:-TLSv1.1:-TLSv1.0:-SSLv3:-SSLv2
SSLProtocol TLSv1.2
SSLCACertificateFile "/etc/httpd/conf/kirdeika-tcc.pem"
ServerName serv_a.kirdeika.org:443
ServerAdmin kirdeikajr@gmail.com
```

Então, habilita-se e inicia-se o serviço *httpd* com os comandos a seguir.

```
# systemctl enable httpd
# systemctl start httpd
```

O *download* do LAM pode ser realizado a partir do *site* do projeto, em <https://www.ldap-account-manager.org/lamcms/releases>. Conforme o manual do LAM(2015), descompacta-se o arquivo baixado, altera-se as configurações definindo locais, permissões e cópias de arquivos de configuração, realizadas com os comandos a seguir.

```
# pacman -s wget
# wget http://prdownloads.sourceforge.net/lam/ldap-account-manager-
5.0.tar.bz2?download
# mv ldap-account-manager-5.0.tar.bz2\?download ldap-account-manager-
5.0.tar.bz2
# tar xjf ldap-account-manager-5.0.tar.bz2 -c /srv/http/
# mv /srv/http/ldap-account-manager-5.0 /srv/http/lam
# cp /srv/http/lam/config/config.cfg.sample /srv/http/lam/config/config.cfg
# cp /srv/http/lam/config/unix.conf.sample /srv/http/lam/config/unix.conf
# chown -r http:http /srv/http/lam
# chmod +x /srv/http/lam/lib/lamdaemon.pl
```

Oberva-se a necessidade de incluir o certificado da autoridade local no navegador a ser utilizado, para que a identidade da conexão possa ser atestada. No navegador Firefox, isto pode ser realizado acessando o Menu → preferências → Avançado → Certificados → Autoridades → Importar, indicando-se o certificado da autoridade e clicando-se em Abrir.

Acessa-se então a página de administração do LAM em <https://serv_a.kirdeika.org/lam>, com um dos navegadores referidos anteriormente, a partir do *HOST*. Prossegue-se com a configuração de acesso utilizando o usuário cn=root,dc=kirdeika,dc=org para o servidor *OpenLDAP* em ldaps://serv_a.kirdeika.org:636, conforme as informações de *login* do manual do LAM.

5 CONFIGURAÇÃO DO ISC BIND

5.1 Instalação

O servidor de nomes a ser instalado deve ser primário, servir como servidor autoritativo para o domínio kirdeika.org, exemplo deste trabalho, e também servir como recursivo, apenas para as máquinas clientes internas da rede. Assim, dados sobre as máquinas internas podem ser fornecidas a outros sistemas, mas a função recursiva deve ser exclusiva às redes internas. Igualmente, a transferência de zona deve ser negada, neste caso. Não haverá necessidade de serem instalados servidores secundários ou escravos, para a solução proposta nestas dimensões, pois a redundância será realizada através da replicação da fonte de dados, o *OpenLDAP*. Será habilitada a função de *update* dinâmico pelo DHCP, que assim incluirá a cada IP fornecido a um cliente seu respectivo nome de domínio totalmente qualificado.

A versão oficial do ISC-BIND (BIND), compilada na distribuição *ArchLinux* não oferece suporte a zonas carregadas dinamicamente, *Dinamically loadable zones*(DLZ) . Para habilitar o suporte ao recurso, essencial para uso do BIND com base de dados em *LDAP*, é necessário recompilar o pacote.

O sistema de pacotes de *software* do *ArchLinux* fornece a ferramenta *abs*, conceitualmente semelhante ao *ports* do *FreeBSD*. Conforme a seção *Arch Build System* do *Wiki* do *ArchLinux*(2015), a compilação é realizada com base em *scripts* utilizados por ferramentas específicas, onde o trabalho de configuração é realizado apenas uma vez, e a compilação refeita a cada revisão do *software* com base nestes *scripts*. Ainda, este modelo permite um controle sobre as alterações, prevenção de atualizações indevidas, versionamento, inclusão de arquivos de configuração específicos, entre outras.

O BIND será recompilado e instalado, com base neste método. Para isso, a ferramenta *abs* deve ser instalada, em seguida a árvore *abs* é criada e o ramo relativo ao BIND é copiado para outro local, onde será personalizado de acordo com

as necessidades deste projeto.

O arquivo de configuração PKGBUILD deve ter as linhas a seguir alteradas.

```
...
pkgname=(bind-dlz bind-tools-dlz)
...
build() {
  cd bind-$_pkgver
  ./configure \
    --prefix=/usr \
    --sysconfdir=/etc \
    --sbindir=/usr/bin \
    --localstatedir=/var \
    --disable-static \
    --with-openssl \
    --enable-threads \
    --with-gssapi=yes \
    --with-libtool \
    --with-libxml2 \
    --with-dlopen=yes \
    --with-dlz-ldap \
    --with-dlz-filesystem=yes \
    --enable-filter-aaaa \
    --enable-rrl \
    --with-ecdsa \
    --enable-threads \
    --with-python=/usr/bin/python \
    --with-geoip \
    --with-ipv6 \
    --with-idn \
    --with-openssl \
    --enable-openssl-hash \
    --enable-filter-aaaa \
    --with-libxml2 \
    --with-libtool
  make
}
...
package_bind-dlz() {
...
      conflicts=('bind')
      depends=('glibc' 'libxml2' 'libcap' 'openssl' 'geoip' 'bind-tools-
dlz')
...
package_bind-tools-dlz() {
...
```

Os comandos a seguir realizam a tarefa de compilação e instalação dos

pacotes *bind-dlz* e *bind-tools-dlz*, originados dos pacotes originais *bind* e *bind-tools* , e os copiam para o outro servidor serv_b, onde devem ser igualmente instalados com o *pacman*.

```
#   pacman -Sy abs python
#   abs extra/bind
#   su – mario
$   cp -R /var/abs/extra/bind .
$   gpg --recv-keys 6FA6EBC9911A4C02
$   cd bind/
$   nano PKGBUILD
$   cp bind.install bind-dlz.install
$   makepkg -f -i
$   exit
#   cd /home/mario/bind
#   pacman -U bind-dlz-9.10.2.P3-1-x86_64.pkg.tar.xz \
      bind-tools-dlz-9.10.2.P3-1-x86_64.pkg.tar.xz
#   scp *.pkg.tar.gz root@serv_b:
```

5.2 Configuração do *OpenLDAP*

Conforme o manual do *driver dlz* para *OpenLDAP* (README), incluído no código fonte do *software*, disponível após a compilação do pacote *bind-dlz* em /home/mario/bind/src/bind-9.10.2-P3/contrib/dlz/modules/ldap/testing, é necessário incluir o arquivo de *schema* na configuração do *OpenLDAP*, e então adicionar os dados das zonas. A conversão pode ser feita com a ferramenta *slaptest*, usando o arquivo de configuração bind.conf com o conteúdo a seguir:

```
include /home/mario/bind/src/bind-9.10.2-
P3/contrib/dlz/modules/ldap/testing/dlz.schema
```

Então, os comandos a seguir criam e incluem no *OpenLDAP* o *schema* necessário:

```
# cd /etc/openldap
# mkdir bind
# cd bind
# slaptest -f bind.conf -F /etc/openldap/bind/
# cp cn\=config/cn\=schema/cn\=\{0\}dlz.ldif dlz.ldif
# sed -i '1,2d' dlz.ldif
# sed -i 's/{0}dlz/dlz/g' dlz.ldif
```

```
# sed -i '/structuralObjectClass\:/,$d dlz.ldif
# sed -i 's/dn\: cn\=dlz/ dn\: cn\=dlz\,cn\=schema\,cn\=config/' dlz.ldif
# ldapadd -D "cn=root,dc=kirdeika,dc=org" -W -f dlz.ldif
```

Então, um usuário específico para o *OpenLDAP* e os objetos restantes necessários são criados com o arquivo bind.ldif, do qual algumas partes estão a seguir. O arquivo completo pode ser visualizado no Apêndice A – Arquivos de configuração deste trabalho:

```
dn: cn=bind,dc=kirdeika,dc=org
objectClass: organizationalRole
cn: bind
description: Gerente do Diretório do DNS
objectClass: simpleSecurityObject
userPassword: {SSHA}y/fnI2RoB+g4GWyrjNn5MiEdUF28pevK

# server suffix - dc=kirdeika,dc=org

dn: ou=dns,dc=kirdeika,dc=org
objectclass: organizationalUnit
ou: dns

dn: dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,dc=org
objectclass: dlzZone
dlzZoneName: kirdeika.org

dn: dlzHOSTName=@,dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,dc=org
objectclass: dlzHOST
dlzHOSTName: @

dn: dlzHOSTName=www,dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,dc=org
objectclass: dlzHOST
dlzHOSTName: www
...
dn: dlzHOSTName=ns1,dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,dc=org
objectclass: dlzHOST
dlzHOSTName: ns1
...
dn: dlzHOSTName=~,dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,dc=org
objectclass: dlzHOST
dlzHOSTName: ~

dn:
dlzRecordID=1,dlzHOSTName=@,dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,dc=
org
objectclass: dlzGenericRecord
dlzRecordID: 1
dlzHOSTName: @
dlzType: txt
dlzData: "this is a text record"
dlzTTL: 10
```

```
dn:
dlzRecordID=2,dlzHOSTName=www,dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,d
c=org
objectclass: dlzARecord
dlzRecordID: 2
dlzHOSTName: www
dlzType: a
dlzIPAddr: 192.168.0.1
dlzTTL: 10
...
dn:
dlzRecordID=14,dlzHOSTName=~,dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,dc
=org
objectclass: dlzARecord
dlzRecordID: 14
dlzHOSTName: ~
dlzType: a
dlzIPAddr: 192.168.0.250
dlzTTL: 10

dn: dlzRecordID=15,dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,dc=org
objectclass: dlzXFR
dlzRecordID: 15
dlzIPAddr: 127.0.0.1
```

As regras de acesso para este usuário são definidas com o arquivo acerto-acessos-dns.ldif a seguir:

```
dn: olcDatabase={0}config,cn=config
changetype: modify
delete: olcAccess
-
add: olcAccess
olcAccess: {0}to *
 by dn.base="gidNumber=0+uidNumber=0,cn=peercred,cn=external,cn=auth"
manage
 by dn.base="cn=root,dc=kirdeika,dc=org" manage
 by dn.base="cn=mario,ou=Usuários,dc=kirdeika,dc=org" manage
 by self write
 by * none
olcAccess: {1}to dn="ou=dns,dc=kirdeika,dc=org"
 by dn.base="cn=bind,dc=kirdeika,dc=org" manage
 by * none
```

Deve-se testar o acesso do usuário "cn=bind,dc=kirdeika,dc=org" para garantir o acesso aos registros do ramo "ou=dns,dc=kirdeika,dc=org" da árvore *LDAP*. Como no comando a seguir:

```
# ldapsearch -D "cn=bind,dc=kirdeika,dc=org" -W -H ldapi:/// -b
ou=dns,dc=kirdeika,dc=org
```

Devem ser retornados todos os registros adicionados a unidade organizacional informada.

Assim como a zona direta, a zona reversa pode ser configurada com o conteúdo do arquivo zona-0.168.192-IN-ADDR.ARPA.ldif, cujo conteúdo está a seguir:

```
dn: cn=dhcp,dc=kirdeika,dc=org
objectClass: organizationalRole
objectClass: simpleSecurityObject
cn: dhcp
description:: R2VyZW50ZSBkbyBEaXJldMOzcmlvIGRvIERIQ1A=
userPassword:: e1NTSEF9eS9mbkkyUm9CK2c0R1d5cmpObjVNaUVkVUYyOHBldks=
structuralObjectClass: organizationalRole

dn: ou=DHCP,dc=kirdeika,dc=org
objectClass: organizationalUnit
ou: dhcp
structuralObjectClass: organizationalUnit

# serv_a.kirdeika.org, DHCP, kirdeika.org
dn: cn=serv_a.kirdeika.org,ou=DHCP,dc=kirdeika,dc=org
objectClass: top
objectClass: dhcpServer
cn: serv_a.kirdeika.org
dhcpServiceDN: cn=DHCP Service Config,ou=DHCP,dc=kirdeika,dc=org

# DHCP Service Config, DHCP, kirdeika.org
dn: cn=DHCP Service Config,ou=DHCP,dc=kirdeika,dc=org
cn: DHCP Service Config
objectClass: top
objectClass: dhcpService
objectClass: dhcpOptions
dhcpPrimaryDN: cn=serv_a.kirdeika.org,ou=DHCP,dc=kirdeika,dc=org
dhcpSecondaryDN: cn=serv_b.kirdeika.org,ou=DHCP,dc=kirdeika,dc=org
dhcpOption: domain-name "kirdeika.org"
dhcpOption: domain-name-servers 192.168.0.1
dhcpOption: routers 192.168.0.100
dhcpOption: netbios-node-type 8
dhcpStatements: ddns-update-style none
dhcpStatements: default-lease-time 86400
dhcpStatements: max-lease-time 86400
dhcpStatements: authoritative
dhcpStatements: log-facility local7
dhcpStatements: on commit {      set ClientName = concat("dhcp-", binary-to-as
 cii(10, 8, "-", leased-address), ".", config-option domain-name);     set Cli
 entIP = binary-to-ascii(10, 8, ".", leased-address);       set ClientPTR = co
 ncat (binary-to-ascii (10, 8, ".", reverse (1, leased-address)),".IN-ADDR.ARP
 A");      execute("/etc/openldap/update_dns.sh", "add", ClientIP, ClientPTR,
 ClientName);  }
dhcpStatements: on expiry  {set ClientIP = binary-to-ascii(10, 8, ".", lease
 d-address) ;  execute ("/etc/openldap/update_dns.sh", "delete", ClientIP);
  }
dhcpStatements: on release {set ClientIP = binary-to-ascii(10, 8, ".", lease
 d-address) ;  execute ("/etc/openldap/update_dns.sh", "delete", ClientIP);
  }
# serv_b.kirdeika.org, DHCP, kirdeika.org
dn: cn=serv_b.kirdeika.org,ou=DHCP,dc=kirdeika,dc=org
cn: serv_b.kirdeika.org
```

```
dhcpServiceDN: cn=DHCP Service Config,ou=DHCP,dc=kirdeika,dc=org
objectClass: top
objectClass: dhcpServer

# 192.168.0.0, DHCP Service Config, DHCP, kirdeika.org
dn: cn=192.168.0.0,cn=DHCP Service Config,ou=DHCP,dc=kirdeika,dc=org
objectClass: top
objectClass: dhcpOptions
objectClass: dhcpSubnet
cn: 192.168.0.0
dhcpNetMask: 24
dhcpOption: domain-name "kirdeika.org"
dhcpOption: netbios-node-type 8
dhcpOption: subnet-mask 255.255.255.0

# RedeLocal, 192.168.0.0, DHCP Service Config, DHCP, kirdeika.org
dn: cn=RedeLocal,cn=192.168.0.0,cn=DHCP Service Config,ou=DHCP,dc=kirdeika,dc=
 org
objectClass: dhcpPool
dhcpRange: 192.168.0.150 192.168.0.200
cn: RedeLocal
```

5.3 Configuração do ISC-BIND

O arquivo named.conf localizado em /home/mario/bind/src/bind-9.10.2-P3/contrib/dlz/modules/ldap/testing/ contém a maior parte da configuração, bastando alterar as definições deste arquivo para a presente instalação, salvando-o em /etc . Os dados sobre a zona atendida virão do *OpenLDAP*, carregadas dinamicamente. Nesta configuração, cada consulta realizada por um cliente resulta em tres consultas ao *OpenLDAP*, e portanto a informação constante neste é fornecida, podendo ser atualizada sempre e quantas vezes for necessário, sem a interrupção do serviço de resolução de nomes. Da mesma forma, a reconfiguração das informações sobre zonas podem ser alteradas dinamicamente, sem interrupções. O campo $senha deve refletir a senha do usuário "cn=bind". O conteúdo do arquivo /etc/named.conf, que define o funcionamento do ISC-BIND, está a seguir:

```
options {
    directory "/var/named";
    pid-file "/run/named/named.pid";
    listen-on-v6 { none; };
    listen-on { any; };
    allow-recursion { internos; };
    allow-transfer { restritos; };
    allow-update { restritos; };
    recursion no;
    version "Indisponivel";
    HOSTname "Invisivel";
    server-id "Nenhum";
};
```

```
logging{
  channel simple_log {
    file "/var/log/named.log" versions 3 size 5m;
    severity info;
    print-time yes;
    print-severity yes;
    print-category yes;
  };
  category default{
    simple_log;
  };
};
acl internos {  172.16.0.0/24; 192.168.0.0/24; 127.0.0.1; };
acl externos {  any; };
acl restritos { 172.16.0.100; 172.16.0.10; 172.16.0.20; 127.0.0.1; };
zone "." IN { type hint; file "root.hint"; };
key "serv_a.kirdeika.org" {
       algorithm hmac-sha256;
       secret "+C7shgjBvcrZnxX6GYJeSi1EnDj0u9cghbkx7OfSAX0=";
 };
controls {
      inet 127.0.0.1 port 953
              allow { 127.0.0.1; } keys { "serv_a.kirdeika.org"; };
};
dlz "kirdeika.org" {
        database "LDAP 2 v3 simple
        {cn=bind,dc=kirdeika,dc=org} {$senha} {127.0.0.1}
        ldap:///dlzZoneName=$zone$,ou=dns,dc=kirdeika,dc=org???
objectclass=dlzZone
ldap:///dlzHOSTName=$record$,dlzZoneName=$zone$,ou=dns,dc=kirdeika,dc=org?
dlzTTL,dlzType,dlzPreference,dlzData,dlzIPAddr?sub?
(&(objectclass=dlzabstractRecord)(!(dlzType=soa)))
        ldap:///dlzHOSTName=@,dlzZoneName=$zone$,ou=dns,dc=kirdeika,dc=org?
dlzTTL,dlzType,dlzData,dlzPrimaryNS,dlzAdminEmail,dlzSerial,dlzRefresh,dlzR
etry,dlzExpire,dlzMinimum?sub?(&(objectclass=dlzabstractRecord)
(dlzType=soa))
        ldap:///dlzZoneName=$zone$,ou=dns,dc=kirdeika,dc=org?
dlzTTL,dlzType,dlzHOSTName,dlzPreference,dlzData,dlzIPAddr,dlzPrimaryNS,dlz
AdminEmail,dlzSerial,dlzRefresh,dlzRetry,dlzExpire,dlzMinimum?sub?
(&(objectclass=dlzabstractRecord)(!(dlzType=soa)))
        ldap:///dlzZoneName=$zone$,ou=dns,dc=kirdeika,dc=org??sub?
(&(objectclass=dlzXFR)(dlzIPAddr=$client$))";
};
```

O named (ISC_BIND) é controlado pela ferramenta *rndc*, cujo acesso ao *named* é possibilitado pela chave de acesso serv_a.kirdeika.org nomeada acima. Para a sua criação, e observando a boa prática de alterar todas as senhas padrão fornecidas com *softwares* instalados, são necessários os comandos a seguir, que criam uma nova chave e um arquivo de configuração de senha e controle que pode ser incluído na configuração do *named*.

```
# rndc-confgen -a -A hmac-sha256 -k serv_a.kirdeika.org -p 953 -s 127.0.0.1
-u named
```

Este comando produz o resultado listado a seguir, com cada parte correspondente incluída em seu devido arquivo. Assim, a ferramenta *rndc* pode controlar o *named,*com a chave serv_a.kirdeika.org incluída nos arquivos /etc/rndc.conf e /etc/named.conf:

```
# Start of rndc.conf
key "serv_a.kirdeika.org" {
        algorithm hmac-sha256;
        secret "+C7shgjBvcrZnxX6GYJeSi1EnDj0u9cghbkx7OfSAX0=";
};

options {
        default-key "serv_a.kirdeika.org";
        default-server 127.0.0.1;
        default-port 953;
};
# End of rndc.conf

# Use with the following in named.conf, adjusting the allow list as needed:
# key "serv_a.kirdeika.org" {
#       algorithm hmac-sha256;
#       secret "+C7shgjBvcrZnxX6GYJeSi1EnDj0u9cghbkx7OfSAX0=";
# };
#
# controls {
#       inet 127.0.0.1 port 953
#               allow { 127.0.0.1; } keys { "serv_a.kirdeika.org"; };
# };
# End of named.conf
```

6 CONFIGURAÇÃO DO ISC-DHCP

6.1 Instalação

O pacote dhcp fornecido pela distribuição *ArchLinux* não contempla a funcionalidade de acesso a bases de dados *LDAP*. Assim, da mesma forma que o pacote foi reconstruído para o ISC-BIND, o ISC-DHCP será reconstruído. Os comandos a seguir mostram as ações necessárias.

```
# abs extra/dhcp
# su – mario
$ cp -R /var/abs/extra/dhcp .
$ cd dhcp
```

Após a atualização do pacote extra/dhcp do *abs,* o arquivo PKGBUILD final é copiado para o usuário comum para que possa ser recompilado, ficando com as seções alteradas conforme os trechos a seguir .

```
...
pkgbase=dhcp
pkgname=('dhcp-LDAP' 'dhclient-LDAP')
...
build() {
...
    --with-LDAP \
    --with-LDAPcrypto
  make
}
package_dhcp-LDAP(){
...
  conflicts=('dhcp')
...
package_dhclient-LDAP(){
...
  conflicts=('dhclient')
…
```

O pacote é então recompilado e instalado com o comando “makepkg -i -f”.

Conforme a seção DHCPd do *Wiki* do *ArchLinux*(2015), e segundo as necessidades deste projeto, o *unit* do *systemd* de controle do *dhcpd* deve ser

editado para permitir que o *dhcpd* possa responder a solicitações de endereços por interface, apenas pela interface eth0 do serv_a.

Tendo o DHCP sido instalado anteriormente, o arquivo /usr/lib/systemd/system/dhcpd4@.service, com o conteúdo a seguir deve ser criado, conforme a seção do *Wiki* mencionada:

```
[Unit]
Description=IPv4 DHCP server on %I
Wants=network.target
After=network.target

[Service]
Type=forking
PIDFile=/run/dhcpd4-%I.pid
ExecStart=/usr/bin/dhcpd -4 -q -pf /run/dhcpd4-%I.pid %I
KillSignal=SIGINT

[Install]
WantedBy=multi-user.target
```

O arquivo de configuração /etc/dhcpd.conf fica com o conteúdo a seguir.

```
LDAP-server "localHOST";
LDAP-port 389;
LDAP-username "cn=dhcp,dc=kirdeika,dc=org";
LDAP-password "$senha";
LDAP-base-dn "dc=kirdeika,dc=org";
LDAP-method dynamic;
LDAP-debug-file "/var/log/dhcp-LDAP-startup.log";
```

6.2 Configuração do *OpenLDAP*

Após a recompilação do pacote, o diretório /home/mario/dhcp/src/dhcp-4.3.2/contrib/ldap contém o arquivo README.ldap com as informações necessárias à configuração do *OpenLDAP* e do DHCP. Conforme o mesmo, deve ser incluído no *OpenLDAP* o *schema* do DHCP, adicionados os valores de *index* dos objetos usados nas pesquisas e definidas as seções de configuração correspondentes ao *OpenLDAP* no arquivo /etc/dhcpd.conf.

```
# ldapadd -D "cn=root,dc=kirdeika,dc=org" -W -f dhcpd.ldif
```

6.3 Configuração do ISC-DHCP

Observa-se que os endereços entregues por este servidor, destinam-se primariamente, mas não exclusivamente, a máquinas cujos endereços não sejam fixos. Portanto, máquinas que servem a propósitos específicos, cujos nomes não sofreriam mudanças constantes, pela própria natureza de seus propósitos, como servidores *web*, *e-mail*, aplicações diversas, entre outros, não teriam seus endereços controlados por este serviço DHCP, e estariam fixos, registrados em associações de nomes e endereços no serviço DNS. Assim, a atualização dinâmica de endereços por entregas (*leases*) do DHCP não serviria a estas máquinas, e deveria ser controlada. Para as necessidades deste projeto, foi prevista a atualização dinâmica do DNS pelos *leases* do DHCP, com nomes iniciando por ”dhcp-” seguidos do endereço entregue. Estas atualizações teriam a validade do *lease*, ou seja, assim que o endereço fosse devolvido (*release*) ou expirasse o tempo de *lease (expiry)* , estes endereços seriam deletados do DNS.

Quando um endereço é requisitado ao servidor DHCP, um evento é gerado. Cada situação, como a entrega(*lease*), devolução(*release*) ou expiração(*expiry*) gera seu evento específico, conforme se a máquina solicitante tem seu endereço físico de *hardware* (ou MAC *address*) registrado. Em caso de haver cadastro, os eventos *release* e *expiry* não são gerados. A versão utilizada do servidor DHCP permite que *scritps* sejam executados a cada ocorrência de um evento, automatizando tarefas de administração.

Com a base de dados do DNS no modo DLZ, a atualização direta entre os serviços DHCP e DNS fica impossibilitada por características dos *softwares*, mas devido a função de eventos do DHCP, foi definido pela criação de um *script* específico a ser executado caso os eventos *lease* (*commit)*, *release* ou *expiry* aconteçam. Estes eventos são gerados e consequentemente o *script* é executado, alterando diretamente a base de dados do DNS no *OpenLDAP*.

O algoritmo utilizado e o *script* estão no Apêndice B – *scripts*, deste trabalho, e o mesmo ainda pode ser usado em modo manual para editar os

endereços e nomes do serviço DNS, sem a necessidade de construção de arquivos LDIF específicos.

O arquivo final de configuração do DHCP para o *OpenLDAP* é então adicionado à base de configuração. Porém, o *schema LDAP* relativo ao DHCP deve ser inserido antes, com o comando a seguir.

```
# ldapadd -D "cn=root,dc=kirdeika,dc=org" -W -f dhcp-schema.ldif
```

Neste modo, a configuração do servidor DHCP referente às redes e faixas de endereços a serem entregues a clientes é armazenada diretamente no *OpenLDAP*. Um arquivo LDIF com as configurações necessárias a este projeto tem o conteúdo a seguir:

```
# 192.168.0.0, DHCP Service Config, DHCP, kirdeika.org
dn: cn=192.168.0.0,cn=DHCP Service Config,ou=DHCP,dc=kirdeika,dc=org
objectClass: top
objectClass: dhcpOptions
objectClass: dhcpSubnet
cn: 192.168.0.0
dhcpNetMask: 24
dhcpOption: domain-name "kirdeika.org"
dhcpOption: netbios-node-type 8
dhcpOption: subnet-mask 255.255.255.0

# DHCP, kirdeika.org
dn: ou=DHCP,dc=kirdeika,dc=org
objectClass: organizationalUnit
ou: dhcp

dn: olcDatabase={0}config,cn=config
olcAccess: {1}to dn="ou=dhcp,dc=kirdeika,dc=org" by
dn.base="cn=dhcp,dc=kirdeika,dc=org" manage by * none

# serv_a.kirdeika.org, DHCP, kirdeika.org
dn: cn=serv_a.kirdeika.org,ou=DHCP,dc=kirdeika,dc=org
objectClass: top
objectClass: dhcpServer
cn: serv_a.kirdeika.org
dhcpServiceDN: cn=DHCP Service Config,ou=DHCP,dc=kirdeika,dc=org

# DHCP Service Config, DHCP, kirdeika.org
dn: cn=DHCP Service Config,ou=DHCP,dc=kirdeika,dc=org
cn: DHCP Service Config
objectClass: top
objectClass: dhcpService
```

```
objectClass: dhcpOptions
dhcpPrimaryDN: cn=serv_a.kirdeika.org,ou=DHCP,dc=kirdeika,dc=org
dhcpSecondaryDN: cn=serv_b.kirdeika.org,ou=DHCP,dc=kirdeika,dc=org
dhcpOption: domain-name "kirdeika.org"
dhcpOption: domain-name-servers 192.168.0.1
dhcpOption: routers 192.168.0.100
dhcpOption: netbios-node-type 8
dhcpStatements: ddns-update-style none
dhcpStatements: default-lease-time 86400
dhcpStatements: max-lease-time 86400
dhcpStatements: authoritative
dhcpStatements: log-facility local7
dhcpStatements: on commit { set ClientIP = binary-to-ascii(10, 8, ".",
leased-address); set LeaseDomain = config-option domain-name; set LeaseMask
= binary-to-ascii(10,8,".", config-option subnet-mask);set ClientName =
pick-first-value(config-option host-name, option host-name, host-decl-
name);execute ("/etc/openldap/update_dns.sh", "add", ClientIP, LeaseDomain,
LeaseMask, ClientName);  }
dhcpStatements: on release {set ClientIP = binary-to-ascii(10, 8, ".",
leased-address) ;  execute ("/etc/openldap/update_dns.sh", "delete",
ClientIP);  }
dhcpStatements: on expiry  {set ClientIP = binary-to-ascii(10, 8, ".",
leased-address) ;  execute ("/etc/openldap/update_dns.sh", "delete",
ClientIP);  }
# serv_b.kirdeika.org, DHCP, kirdeika.org
dn: cn=serv_b.kirdeika.org,ou=DHCP,dc=kirdeika,dc=org
cn: serv_b.kirdeika.org
dhcpServiceDN: cn=DHCP Service Config,ou=DHCP,dc=kirdeika,dc=org
objectClass: top
objectClass: dhcpServer
# RedeLocal, 192.168.0.0, DHCP Service Config, DHCP, kirdeika.org
dn: cn=RedeLocal,cn=192.168.0.0,cn=DHCP Service
Config,ou=DHCP,dc=kirdeika,dc=
 org
objectClass: dhcpPool
dhcpRange: 192.168.0.150 192.168.0.200
cn: RedeLocal
```

Este arquivo deve ser adicionado à configuração *LDAP* para que se possa iniciar o servidor DHCP, e os *pools* (faixas de endereços de rede) servidos por este DHCP podem então ser alterados dinamicamente. Então, o serviço pode ser configurado, habilitado e iniciado com os comandos a seguir:

```
# ldapadd -D “cn=root,dc=kirdeika,dc=org” -W -H ldapi:/// -f
/etc/openldap/dhcp/dhcp.ldif
# systemctl enable dhcpd4@eth0
# systemctl start dhcpd4@eth0
```

7 CONFIGURAÇÃO DO *UCARP*

7.1 Instalação

O *software UCARP* já foi instalado em passos anteriores, e nada mais é necessário ao seu funcionamento a não ser especificar os parâmetros necessários em linha de comando, alinhados com as necessidades deste trabalho.

7.2 Configuração do *SYSLOG-NG*

Em virtude da necessidade de que os eventos gerados pelo funcionamento do *UCARP* sejam registrados, e pela opção de linha de comando em que os eventos possam ser direcionados a uma facilidade local do *syslog-ng*, basta que sejam informados os parâmetros de filtro e arquivo ao *syslog-ng*. As seções a serem adicionadas ao arquivo de configuração de cada servidor da solução, em /etc/syslog-ng/syslog-ng.conf, estão a seguir:

```
template template_date_format {
    template("${YEAR}/${MONTH}/${DAY} ${HOUR}:${MIN}:${SEC} ${HOST} $
{MSGHDR}${MSG}\n");
    template_escape(no);
};

filter f_local5         {facility(local5); };
destination ucarp { file("/var/log/ucarp.log"
template(template_date_format)); };
log {source(src); filter(f_local5); destination(ucarp); };
```

Como pode ser visto nas linhas acima, o *syslog* filtra os eventos vindos do *UCARP*, e os armazena no arquivo /var/log/ucarp.log. Esta automação será essencial para os *scripts* de automação da alta disponibilidade. Então, o *syslog* deve ser reiniciado para aplicar as configurações, conforme o comando a seguir, aplicado nos dois servidores (serv_a e serv_b):

```
# systemctl restart syslog-ng
```

7.3 Inicialização do *UCARP* no serv_a

Uma vez que o *syslog-ng* esteja preparado, o *UCARP* pode ser iniciado e o *log* analisado para a verificação. Observa-se que o arquivo de *log* /var/log/ucarp.log apenas será criado quando da primeira inicialização do *UCARP*. O comando, aplicado no serv_a, inicia o *UCARP* em modo *MASTER*, aplicando-se neste momento as premissas que o serv_a terá prioridade na solução, o endereço do *cluster* de alta disponibilidade será 1 e o endereço de rede a ter o servidor eleito a responder será 192.168.0.1. Já o comando aplicado no serv_b, inicia o *UCARP* já em modo *backup*, com prioridade menor, mesmo endereço de cluster e endereço de rede a ser tratado. A página de manual do *UCARP* contém as explicações das opções. No capítulo 8 será descrito o algoritmo criado para a inteligência da solução de alta disponibilidade deste trabalho, onde as definições aplicáveis estarão adaptadas. O comando para o serv_a fica como a seguir:

```
# ucarp -i eth0 -s 192.168.0.10 -v 1 -p 12345seis -P -a 192.168.0.1 -b 1 -k
10 -u /usr/bin/vip-up.sh -d /usr/bin/vip-down.sh -B -f local4
```

O *UCARP* prevê a execução de dois *scripts*, um em caso de promoção do nó do *cluster*, ou seja, a instância em execução no servidor, é promovida a *Master*. O outro, em caso de despromoção, ou seja, quando a instância é rebaixada a *backup* ou a instância recebe o sinal de sistema *SIGTERM* ou *SIGKILL*. Para as necessidades deste trabalho, tais *scripts* controlarão a configuração do endereço de rede a ser compartilhado pelos nós da solução e enviarão *e-mails* sobre as mudanças. Tais *scripts* estão no Apêndice C – *scripts*. Uma vez iniciado o *UCARP* em serv_a, as entradas a seguir podem ser visualizadas no arquivo de *log* do *UCARP*, /var/log/ucarp.log:

```
Aug 21 14:33:37 serv_a.kirdeika.org UCARP[5296]: [INFO] Local advertised
ethernet address is [08:00:27:4f:64:de]
Aug 21 14:33:38 serv_a.kirdeika.org UCARP[5296]: [WARNING] Switching to
state: backup
Aug 21 14:33:38 serv_a.kirdeika.org UCARP[5296]: [WARNING] Spawning
[/usr/bin/vip-down.sh eth0 192.168.0.1]
Aug 21 14:33:42 serv_a.kirdeika.org UCARP[5296]: [WARNING] Switching to
state: Master
Aug 21 14:33:42 serv_a.kirdeika.org UCARP[5296]: [WARNING] Spawning
[/usr/bin/vip-up.sh eth0 192.168.0.1]
```

7.4 Inicialização do *UCARP* no serv_b

A inicialização de teste do *UCARP* em modo *backup*, com prioridade menor no serv_b segue o comando a seguir.

```
# ucarp -i eth0 -s 192.168.0.20 -v 1 -p 12345seis -a 192.168.0.1 -b 1 -k 50
-u /usr/bin/vip-up.sh -d /usr/bin/vip-down.sh -B -f local5
```

Então, pode ser visualizada a inicialização do *software* no arquivo de *log*, conforme as entradas a seguir.

```
Aug 21 15:21:02 serv_b.kirdeika.org UCARP[414]: [INFO] Local advertised
ethernet address is [08:00:27:ee:34:82]
Aug 21 15:21:02 serv_b.kirdeika.org UCARP[414]: [WARNING] Switching to
state: backup
Aug 21 15:21:02 serv_b.kirdeika.org UCARP[414]: [WARNING] Spawning
[/usr/bin/vip-down.sh eth0 192.168.0.1]
```

Pode ser vista a troca de mensagens entre os servidores, conforme o monitoramento de pacotes extraído na interface eth0 do serv_b. Nesta sequencia, observa-se a mudança na eleição na rede, quando o servidor serv_a(192.168.0.10), prioridade 10, deixa de enviar os avisos a rede a partir do instante 15:40:48.966540, sendo precedido pelo serv_b(192.168.0.20), no instante 15:40:52.120532. Este é o momento em que o servidor serv_b é promovido a *Master*, como pode ser visto no *log* em sequencia. Foi extraída a sequencia de pacotes com *tcpdump* ("tcpdump -l -n -i eth0 -vvv net 224.0.0.0/4") e demonstrada a seguir:

```
15:40:46.926485 IP (tos 0x10, ttl 255, id 37483, offset 0, flags [DF],
proto VRRP (112), length 56)
    192.168.0.10 > 224.0.0.18: vrrp 192.168.0.10 > 224.0.0.18: VRRPv2,
Advertisement, vrid 1, prio 10, authtype none, intvl 1s, length 36,
addrs(7):
115.17.113.81,121.5.141.215,71.126.242.65,250.146.248.144,252.254.145.140,1
2.191.122.231,98.51.65.243
15:40:47.966963 IP (tos 0x10, ttl 255, id 60324, offset 0, flags [DF],
proto VRRP (112), length 56)
    192.168.0.10 > 224.0.0.18: vrrp 192.168.0.10 > 224.0.0.18: VRRPv2,
Advertisement, vrid 1, prio 10, authtype none, intvl 1s, length 36,
addrs(7):
115.17.113.81,121.5.141.216,203.90.198.223,153.176.239.112,40.15.122.233,23
1.114.10.216,161.204.255.246
15:40:48.746321 IP (tos 0xc0, ttl 1, id 0, offset 0, flags [DF], proto IGMP
(2), length 40, options (RA))
    192.168.0.10 > 224.0.0.22: igmp v3 report, 1 group record(s) [gaddr
```

```
224.0.0.18 to_in { }]
15:40:48.966540 IP (tos 0xc0, ttl 1, id 0, offset 0, flags [DF], proto IGMP
(2), length 40, options (RA))
    192.168.0.10 > 224.0.0.22: igmp v3 report, 1 group record(s) [gaddr
224.0.0.18 to_in { }]
15:40:52.120532 IP (tos 0x10, ttl 255, id 52570, offset 0, flags [DF],
proto VRRP (112), length 56)
    192.168.0.20 > 224.0.0.18: vrrp 192.168.0.20 > 224.0.0.18: VRRPv2,
Advertisement, vrid 1, prio 50, authtype none, intvl 1s, length 36,
addrs(7):
115.17.113.81,121.5.141.217,203.205.208.58,142.117.117.102,10.39.97.85,3.13
6.82.202,248.119.222.47
15:40:53.298068 IP (tos 0x10, ttl 255, id 43380, offset 0, flags [DF],
proto VRRP (112), length 56)
    192.168.0.20 > 224.0.0.18: vrrp 192.168.0.20 > 224.0.0.18: VRRPv2,
Advertisement, vrid 1, prio 50, authtype none, intvl 1s, length 36,
addrs(7):
115.17.113.81,121.5.141.218,174.138.188.48,123.100.244.135,52.131.233.161,1
36.227.202.133,225.99.124.222
15:40:54.494689 IP (tos 0x10, ttl 255, id 19572, offset 0, flags [DF],
proto VRRP (112), length 56)
    192.168.0.20 > 224.0.0.18: vrrp 192.168.0.20 > 224.0.0.18: VRRPv2,
Advertisement, vrid 1, prio 50, authtype none, intvl 1s, length 36,
addrs(7):
115.17.113.81,121.5.141.219,145.132.188.148,56.143.245.208,83.139.100.234,1
82.184.193.100,171.20.242.16
```

Arquivo de *log* no momento da eleição e promoção de serv_b a *Master*.

```
Aug 21 15:40:52 serv_b.kirdeika.org UCARP[414]: [WARNING] Switching to
state: Master
Aug 21 15:40:52 serv_b.kirdeika.org UCARP[414]: [WARNING] Spawning
[/usr/bin/vip-up.sh eth0 192.168.0.1]
```

8 POLÍTICAS PARA A SOLUÇÃO DE ALTA DISPONIBILIDADE

A instalação individual dos *softwares* selecionados não implica que estes estarão sempre funcionais, que os serviços oferecidos estarão sempre disponíveis ou que existam cópias utilizáveis das informações necessárias à recriação dos serviços. Sob este ponto de vista, a solução proposta não pode ser considerada como alta disponibilidade neste momento, e como mencionado no capítulo 1, seção 1.5, as duas técnicas de tolerância a falhas mencionadas por Weber (2003, p.16), de mascaramento e detecção, localização e reconfiguração, devem ser ainda implementadas.

Desta forma, são necessários *scripts* que serão executados periodicamente, realizando testes individualmente nos serviços, reconfigurando a solução em caso de anormalidades. Para estes, as seguintes políticas ficam estabelecidas:

- o servidor serv_a será considerado titular na solução;
- o servidor serv_b será considerado *backup* nesta solução;
- o endereço de rede 192.168.0.1 será divulgado aos clientes como endereço titular dos serviços, através dos padrões no serviço DHCP;
- o nome de máquina domain.kirdeika.org será associado a este endereço no DNS;
- a verificação será realizada a cada 1 minuto;
- um servidor apenas poderá fazer parte do nó de alta disponibilidade se todos os seus serviços estiverem disponíveis;
- em caso de falha em qualquer dos *softwares* da solução, seja ele o próprio *UCARP*, o *OpenLDAP*, o ISC-DHCP ou o ISC-BIND, o *software* correspondente deverá ser reiniciado uma vez, e em caso de nova falha o servidor em questão deverá ser retirado do *cluster*, passando o endereço de rede para o servidor ativo;
- em caso de falha em algum serviço, um aviso detalhado deverá ser enviado por *e-mail* ao administrador da solução;
- sempre que houver mudança na condição do servidor responsável pelos

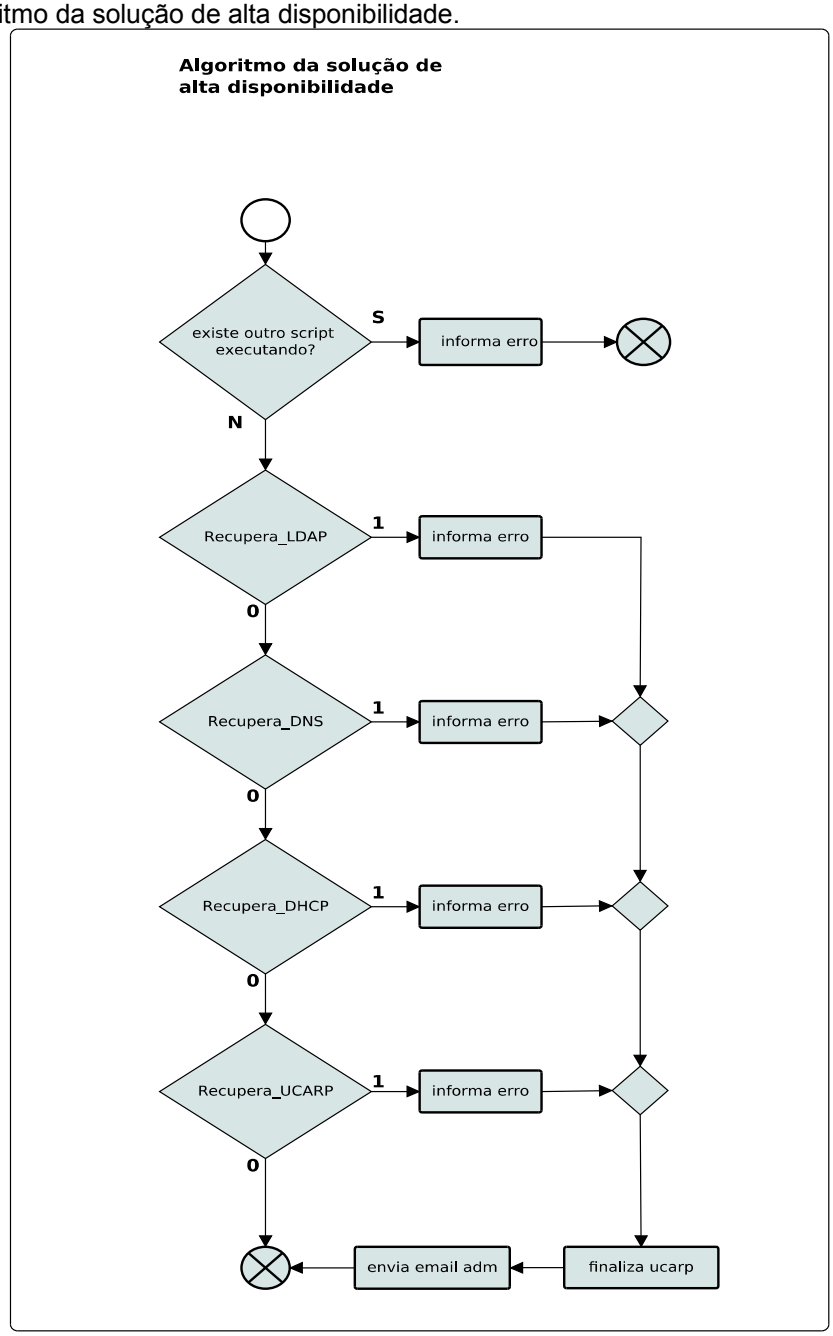
serviços, um aviso deverá ser enviado ao administrador;

- os servidores são entidades independentes, assim, em caso de segmentação da rede, ficando os mesmos isolados entre si, é aceitável que cada um assuma a titularidade em sua porção de rede, até que a falha na rede seja corrigida. Assume-se que a sincronização do *OpenLDAP* estará degradada, e que avisos sobre isso a partir do servidor isolado no segmento de rede em falha não serão possíveis. Para este caso, serão considerados os avisos de falha na replicação do servidor a partir da porção de rede regular;
- deverá haver rotação nos arquivos de *log* sempre que os mesmo atingirem 5Mb ou 1 semana;
- não será implementado neste projeto o monitoramento de disponibilidade de recursos de máquina nos servidores, como memória livre, carga de processamento ou uso de disco e rede. Tais recursos são indispensáveis em termos de segurança, mas pela complexidade envolvida e escopo deste trabalho este monitoramento deve fazer parte de estudo específico, e o autor recomenda o uso do *software Zenoss Core*, atualmente na versão 5, disponível em <http://wiki.zenoss.org> . Alternativas são o *Nagios, OpenNMS, Cacti, Monit, Zabbix*, entre vários outros.

Por consequencia, fica estabelecido que um *script* principal, nomeado domain.sh, será executado periodicamente, chamando os *scripts* que realizarão os testes individuais de cada serviço. Em caso de falha recuperável, estes *scripts* de teste de cada serviço realizarão a tarefa, retornando um código de resultado ao principal. Este código será avaliado, e em caso de falha não recuperável, retirará o nó (servidor em que o teste está sendo executado) do *cluster*. Todos os *scripts* construídos estão em /root/ha nos servidores.

Os algoritmos da solução e os *scripts* resultantes construídos estão descritos no Apêndice C – *scripts*. A solução de alta disponibilidade pode ser entendida pelo algoritmo a seguir, do qual resulta o *script* domain.sh, executado a cada período definido nas políticas.

Figura 4 - Algoritmo da solução de alta disponibilidade.



Fonte: Autor.

9 POLÍTICA DE *BACKUP*

A solução apresentada neste trabalho deve prover um sistema de *backup* confiável, replicável, que forneça material prontamente identificável e utilizável para casos de necessidade.

Por tratar de informações que podem ser dinamicamente alteradas, e que portanto falhas tanto dos sistemas envolvidos quanto dos ativos humanos podem causar problemas, em quaisquer dos pilares da segurança, a extração dos dados e das configurações é premissa para manter a segurança da solução.

Em princípio, devido a natureza das informações e a grandeza relativa dos dados e configurações (tamanho dos arquivos) nesta aplicação, permitem que todos estes dados e configurações possam ter sua salvaguarda realizada em muito curto período de tempo. Por isso, nesta solução de *backup*, define-se que todo o trabalho será realizado de hora em hora, no minuto zero, guardando-se o material gerado local e remotamente. Em aplicações futuras da solução, estes tempos podem ser ajustados conforme a necessidade. Em caso de necessidade, a pesquisa de um determinado ponto no tempo a partir do qual ocorreu uma falha, seja ela provocada ou não, deve permitir a restauração dos dados eventualmente perdidos ou corrompidos, em caso de dano catastrófico à informação ou à solução.

Vale lembrar que por se tratar de um *cluster*, em que a fonte de informações é consistente ao longo dos N servidores *OpenLDAP* em modo *MULTI-MASTER*, a não ser que a falha desabilite totalmente o servidor envolvido, pode ocorrer a replicação da falha na informação pelo *cluster*. A única forma de restaurar é com a utilização da cópia consistente da informação de um determinado momento no passado. Essa é a principal utilização do *backup*.

Como salvaguarda, e aplicado nesta solução, o *backup* a partir de no mínimo dois servidores da solução é recomendado, diminuindo consideravelmente a probabilidade de não haver ao menos um *backup* confiável em caso de necessidade.

Evidentemente, a periodicidade da extração e guarda das informações deve ser revista, assim como o método de extração, em caso de maior quantidade de informações. Nesta solução é aplicado o método de cópia integral, mas metodologias como *backup* diferencial e incremental podem ser analisadas em necessidade futura, sob a ótica da quantidade de informações a ser salva.

Define-se que a extração será realizada a cada minuto zero de cada hora, dos dados e configurações do *OpenLDAP*, salvos em arquivo único sob uma estrutura de diretórios específica. Diariamente, à hora zero, será realizada uma extração do conteúdo anterior, e que conterá mais os *scripts* e pacotes de *software* recompilados da solução, para facilitar a reimplantação integral.

Define-se que o local a receber os arquivos finais de *backup* será /var/backup, e um diretório temporário a ser utilizado é o padrão da distribuição, em /tmp. Deve-se excluir qualquer arquivo temporário após a finalização do *script*.

Todo arquivo de *backup* gerado será criptografado com um o uso de uma chave simétrica, que será encriptada usando uma chave pública, gerada exclusivamente para este propósito. A chave privada, necessária para decriptação da chave simétrica ficará sob guarda do administrador. Este método é válido para quaisquer volumes de *backup*. O uso direto da chave pública na encriptação só permite arquivos de entrada de no máximo tamanho igual ao do arquivo a ser encriptado. No método escolhido, não há esta limitação. Maiores informações a respeito da metodologia utilizada pode ser obtida no artigo “Utilização de criptografia simétrica e assimétrica com *software* livre”, do autor, disponível em <https://drive.google.com/file/d/0B9YwgMsiyRCvRHA4VzBtXzYzMnc/view?usp=sharing>.

O algoritmo e *scripts* resultantes estão descritos no Apêndice V. A configuração do agendador de tarefas cron tem adicionada a linha a seguir.

```
00 * * * * /root/ha/backup.sh 2>&1 >/dev/null
```

10 TESTES E ENSAIOS DE DESEMPENHO

Os testes devem validar todas as políticas aplicadas, atestando o objetivo de se conseguir alta disponibilidade. Uma falha, mesmo que simulada, não pode impedir o funcionamento da solução, devendo o conjunto se reconfigurar mascarando a falha e avisando um administrador por *e-mail*, se possível.

Assim, os testes devem cobrir os objetivos, a saber:

1. fornecimento de endereços de rede e rotas;
2. resolução de nomes;
3. autenticação;
4. teste da alta disponibilidade;
5. teste da replicação do *OpenLDAP*;
6. teste de desempenho de inserção de registros no *OpenLDAP*;
7. teste de desempenho de consultas ao *OpenLDAP*.

Estes testes serão realizados a partir de uma outra máquina virtual, cadastrada na solução, para que a ela seja fornecido um endereço e seu nome associado a seu endereço automaticamente. É parte do protocolo DHCP que a máquina possa informar seu nome ao servidor, e este nome deverá ser adicionado ao resolvedor de nomes. Isto será verificado.

Foi instalada uma nova máquina virtual com *ArchLinux* para os testes. Foi definida uma configuração mínima para a mesma, e apenas os pacotes necessários aos testes foram instalados, e duas conexões de rede configuradas, tendo apenas a interface eth0 configurada para receber endereço por DHCP. Esta máquina foi cadastrada na seção DHCP da interface de administração instalada *Ldap Account Manager*(LAM), tendo sido informados o endereço físico da interface virtual criada e o nome da máquina. Note-se que este nome pode ser usado para cadastro no resolvedor de nomes(DNS), mas a preferência na configuração é para o nome informado pela máquina.

10.1 Teste de fornecimento de endereço e rotas

Esta máquina foi configurada para receber endereço de rede a partir de um servidor DHCP da rede, conforme o *Wiki* do *ArchLinux*, página *Network_configuration*, seção 5.1, *systemd-networkd*(2015).

O reinício da máquina após a configuração foi bem sucedido, a máquina inicializou e obteve endereço de rede e rota padrão, conforme as configurações do DHCP, o êxito e a simplicidade do teste devem-se à natureza intrínseca do protocolo DHCP. Este deve apenas fornecer, quando solicitado, um endereço de rede, rotas e outras configurações relativas à rede, na forma do protocolo, o que se sucedeu com êxito.

10.2 Teste de autenticação

A configuração básica da nova máquina virtual de teste, nomeada ca, foi realizada sem a criação de usuários não privilegiados locais, em virtude da realização do teste do sistema de autenticação provido pelo *LDAP*. Assim, para esta funcionalidade foram seguidas as instruções de configuração para a máquina cliente ca a partir do *Wiki* do *ArchLinux*, página *LDAP_authentication*, seções *Client setup*, NSS Configuration, PAM Configuration, *Create home folders at login* e *Online and Offline Authentication with SSSD*(2015). A primeira autenticação realizada foi bem sucedida e criou o diretório do usuário.

Para avaliar a autenticação em *cache*, foram desligados os servidores serv_a e serv_b, realizado *logout* na máquina ca. Um novo *login* foi realizado com sucesso, mas ao reinicializar a máquina ca, não foi mais possível realizar novo *login* com o usuário comum, cadastrado no *LDAP*. Esta situação demonstra a importância da solução de alta disponibilidade, pois não apenas não havia a possibilidade de usuário comum realizar *login,* como não havia mais acesso à rede, pois não havia servidor DHCP para prover configuração de rede à máquina.

Ao ser reinicializado o serv_a, foi possível notar que a máquina ca

brevemente obteve configuração de rede, e foi então possível a autenticação.

O comando “getent passwd *usuário*” retorna a identificação do usuário informado, como *shell*, diretório *home* e id, o que pode ser usado para testes durante o processo de configuração.

10.3 Teste de resolução de nomes

A resolução de nomes foi bem sucedida, tanto na condição de servidor recursivo, quando foi usada a funcionalidade para resolver os nomes dos servidores externos necessários à instalação dos pacotes, quanto na função autoritativo, pois foi necessária a resolução do endereço do servidor domain.kirdeika.org, configurado como servidor primário de autenticação. Note-se que este nome de máquina refere-se à maquina que detém o endereço principal gerenciado pelo *UCARP*, 192.168.0.1, que em condições normais estará configurado na máquina serv_a, e em caso de problemas será automaticamente configurado na máquina serv_b. Importante lembrar que o endereço acima mencionado é adicionado à interface, não substituindo o endereço inicial da interface, pelo qual ainda se pode alcançar a referida máquina, se as condições de funcionamento assim permitirem.

10.4 Teste da alta disponibilidade

Para o próximo teste, serão realizadas requisições ICMP[14] echo ao *HOST* domain, a partir da máquina cliente ca, de forma ininterrupta pelo período suficiente para completar um ciclo de testes. Neste ciclo, será simulada uma falha na conectividade de rede, desconectando alternadamente serv_a e serv_b.

O esperado é que, pelo período configurado no *UCARP* de 1 segundo de transmissão de aviso de *Master*, e 2 segundos de *ratio*, para consideração pelo *UCARP* em serv_b de que o *UCARP* em serv_a está com problemas, espera-se que o *backup* seja promovido a *Master* nestes dois segundos.

14 - *Internet Control Message Protocol,* conforme definido pela RFC 792, disponível em <https://tools.ietf.org/rfc/rfc792.txt>. Acesso em 17 set. 2015.

Foi constatada a mudança da responsabilidade do endereço 192.168.0.1, configurado no servidor serv_a para o serv_b dois segundos após a desconexão virtual da rede da interface eth0 do serv_a, assumindo serv_b a condição de *Master*. Foram recebidos *e-mails* dos servidores, de serv_b durante a mudança para *Master*, e de serv_a quando a rede foi reconectada informando a condição de *backup*. Observa-se que a função de aviso ao administrador foi implementada diretamente nos *scripts* do *UCARP*, pois estes são acionados em caso de qualquer falha.

Foram recebidos ainda diversas outras mensagens, informando as condições iniciais, onde não havia uma efetiva mudança de estado de responsabilidade entre os servidores, estas mensagens podem ser consideradas espúrias e pedem uma mudança na lógica ou na implementação do sistema de avisos, que pode tornar-se muito ruidoso, ou seja, com mensagens desnecessárias, consequência da implementação realizada nos *scripts* do *UCARP*.

Foi então percebido que a quantidade de requisições ICMP *echo* refletiu os momentos de transição do endereço de rede entre os servidores, mas em 15 minutos de teste, o serv_a não voltou a ser *Master*, que é a condição desejada.

Foram alterados os *scripts* recupera_UCARP.sh em cada servidor. Foi constatado que os parâmetros de inicialização do *UCARP* devem ficar diferenciados, com *Advertisement skew* (-k) com 1 no serv_a e com 50 no serv_b. Ainda, serv_a deve ter adicionado o parâmetro *preemptive failover* (-P). Assim, foi notado que serv_a, após ser retirado do *cluster*, serv_b continua assumindo após 2 segundos, e que serv_a reassume em até 2 minutos após ter o cabo reconectado.

Advertisement skew é configurado no arquivo /etc/domain.conf, parametro PESO, e *preemptive failover* é alterado na própria linha do *script*, mas pode ser adicionado ao final da variável anterior, com um caracter de espaço para o parâmetro “-P”. A linha correspondente do *script* recupera_UCARP.sh fica como a seguir, para serv_a:

```
$UCARP -i $INTERFACE -s $IP_SERVIDOR -v $ID_CLUSTER -p $SENHA -a $IP_CLUSTER -b 1 -k $PESO -u
$SCRIPT_UP -d $SCRIPT_DOWN -r 2 -B -P -z -f $SYSLOG 2>&1 >>$REC_UCARP_LOG
```

10.5 Teste da replicação

Este teste verifica se um registro adicionado/alterado em serv_a é replicado em serv_b. Espera-se que a replicação seja imediata, pois foi configurado o modo de replicação *Multi-Master* do tipo *refreshAndPersist*, ou seja, quando é inicializado o *OpenLDAP*, este abre uma sessão de replicação com o outro *Master*, e cada alteração em registros de um servidor é imediatamente replicada aos outros *Masters*. O ponto de controle da replicação é o parâmetro contextCSN, que em caso de replicação sincronizada é igual nos servidores do nó. Assim, consultando este parâmetro, ou realizando uma consulta do parâmetro alterado no servidor oposto, imediatamente após a sua alteração, atesta-se a replicação.

Para isso, foi criado um *script* que realiza a inserção de um registro de teste, realiza a consulta no servidor oposto e deleta o registro, sem esperas. Repete esses passos quantas vezes forem informadas como parâmetro, e inverte a ordem de servidores, repetindo o ciclo.

O resultado, com 100 ciclos de *insert* em serv_a, *search* em serv_b e *delete* de serv_a, e após invertendo os servidores, foi o mesmo, não houveram erros e o registro foi localizado todas as vezes. O *script* desenvolvido para o teste está a seguir.

```
#!/bin/bash
if [ $# -lt 1 ] ; then
        echo "nao foram informadas quantas iterações, realizando 20"
        sleep 1
         total=20
else
        total=$(($1*2))
fi
count=0
erros_replica=0
for (( i=1; i <= $1; i++ )) ; do
clear
ldapadd -D "cn=root,dc=kirdeika,dc=org" -w $senha -H "ldaps://serv_a.kirdeika.org:636/" -f
insert_teste.ldif 2>&1 >/dev/null
res1=$?
erros_add=$(($erros_add+$res1))
```

```
res2=$(ldapsearch -D "cn=root,dc=kirdeika,dc=org" -w $senha -H
"ldaps://serv_b.kirdeika.org:636/" -b "dc=kirdeika,dc=org" "(&(objectClass=dlzARecord)
(dlzIPAddr=192.168.0.140))" dlzHOSTName | grep teste | grep -v ^# | wc -l)
if [ $res2 -eq 0 ] ; then erros_replica=$(($erros_replica+1)) ; fi
ldapdelete -D "cn=root,dc=kirdeika,dc=org" -w $senha -H "ldaps://serv_a.kirdeika.org:636/"
'dlzRecordID=1,dlzHOSTName=teste,dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,dc=org'
'dlzHOSTName=teste,dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,dc=org'
res3=$?
erros_delete=$(($erros_delete+$res3))
count=$(($count+1))
echo $count
done
for (( i=1 ; i <= $1 ; i++ )) ; do
clear
ldapadd -D "cn=root,dc=kirdeika,dc=org" -w $senha -H "ldaps://serv_b.kirdeika.org:636/" -f
insert_teste.ldif 2>&1 >/dev/null
res1=$?
erros_add=$(($erros_add+$res1))
res2=$(ldapsearch -D "cn=root,dc=kirdeika,dc=org" -w $senha -H
"ldaps://serv_a.kirdeika.org:636/" -b "dc=kirdeika,dc=org" "(&(objectClass=dlzARecord)
(dlzIPAddr=192.168.0.140))" dlzHOSTName | grep teste | grep -v ^# | wc -l)
if [ $res2 -eq 0 ] ; then erros_replica=$(($erros_replica+1)) ; fi
ldapdelete -D "cn=root,dc=kirdeika,dc=org" -w $senha -H "ldaps://serv_b.kirdeika.org:636/"
'dlzRecordID=1,dlzHOSTName=teste,dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,dc=org'
'dlzHOSTName=teste,dlzZoneName=kirdeika.org,ou=dns,dc=kirdeika,dc=org'
res3=$?
erros_delete=$(($erros_delete+$res3))
count=$(($count+1))
echo $count
done
echo "Resultado final , $(($1 * 2)) iterações"
echo "Soma dos erros :  add = $erros_add , replica = $erros_replica , delete : $erros_delete"
```

10.6 Teste de desempenho de inserção de registros no *OpenLDAP*

Para este teste serão inseridos dados no diretório "ou=dns,dc=kirdeika,dc=org", relativos a máquinas fictícias, de endereços sequenciais IPv4 de rede 10.0.0.0/16, de 10.0.0.1 a 10.0.255.254, totalizando 65534 *HOST*S com nomes de máquinas compostos por 10 caracteres aleatórios, no domínio teste.kirdeika.org.

Será contabilizado o tempo em que a tarefa for realizada, que será referente à inserção de 4 registros *LDAP* por máquina, que são relativos a nome de máquina, endereço IP, nome reverso e endereço reverso. Serão incluídos 256 zonas reversas, cada zona com 4 registros. Calcula-se o total de registros *LDAP* em 262.400. Será aferido o total inicial e o final de registros. Para a tarefa, uma versão modificada do *script* update_dns.sh será utilizado, onde apenas as inserções sequenciais serão realizadas, por meio de *loops* nos octetos correspondentes no IP, gerando-se um nome de máquina aleatório de 10 caracteres.

O *script* a seguir, derivado de update_dns.sh, realizou a tarefa.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# versão modificada do update_dn, para inserção automática de registros no dns, para efeitos
de teste
import sys,os,logging,random,string,datetime,ssl
from LDAP3 import Server, Connection, LEVEL, BASE, SUBTREE, ALL, Tls, LDAPException
# globais
total_registros = 0
erros_LDAP = 0
erros_ssl = 0
def random_char(y):
       return ''.join(random.choice(string.ascii_letters) for x in range(y))
def main():
 global total_registros
 global erros_LDAP
 global erros_ssl
 # padrões deste script
 BaseDN = 'ou=dns,dc=kirdeika,dc=org'
 Usuario = 'cn=bind,dc=kirdeika,dc=org'
 HOST = 'domain.kirdeika.org'
 Senha = '$senha'
 CERTIFICADO_CA = '/root/rootCA.pem'
 # configurações TLS
 tls = Tls(validate=ssl.CERT_REQUIRED, version=ssl.PROTOCOL_TLSv1,
ca_certs_file=CERTIFICADO_CA)
 tempo_inicio = datetime.datetime.now()
 Conexao_LDAP = Conecta(HOST,Usuario,Senha,tls)
 B=0
 for C in range(0,255):
    for D in range(1,254):
          IP = "10." + str(B) + "." + str(C) + "." + str(D)
          popula("add",IP,"teste.kirdeika.org","255.255.0.0",random_char(10),Conexao_LDAP)
 Desconecta(Conexao_LDAP)
 tempo_fim = datetime.datetime.now()
 tempo_total = tempo_fim - tempo_inicio
 print("Total de %s inserções LDAP" % (str(total_registros)))
 print("Erros LDAP                          = " + str(erros_LDAP))
 print("Erros SSL                          = " + str(erros_ssl))
 print("Tempo total                         = " + str(tempo_total))
 sys.exit(0)
def popula(Mode,End,Dom,Mask,Hst,Conexao_LDAP):
 global total_registros
 global erros_LDAP
 global erros_ssl
 # texto mostrado caso seja encontrado erro de parametros
 USO = """\n===== Uso: update_dns.sh Modo IP Dominio Mascara_subrede [Nome]"""
 Modo = str(Mode)
 IP = str(End)
 if Modo == 'add' :
        Dominio = str(Dom).strip()
        Mascara_subrede = str(Mask).strip()
        Dominio_reverso = IP.split(".")[2] + "." + IP.split(".")[1] + "." + IP.split(".")[0] +
".IN-ADDR.ARPA"
        Nome_curto_reverso = IP.split(".")[3]
        Nome_reverso = IP.split(".")[3] + "." + Dominio_reverso
        Nome_curto = str(Hst)
        Nome = Nome_curto + "." + Dominio
 # ação
 if Modo == 'add':
    Nome_curto = str(Nome).split('.', maxsplit=1)[0]
    Zona_direta = str(Nome).split('.', maxsplit=1)[1]
    DN_novo_direto = "dlzHOSTName=" + Nome_curto + ",dlzZoneName=" + Zona_direta +
```

```
",ou=dns,dc=kirdeika,dc=org"
    DN_novo_direto_1 = "dlzRecordID=1,dlzHOSTName=" + Nome_curto + ",dlzZoneName=" +
Zona_direta + ",ou=dns,dc=kirdeika,dc=org"
    # montando registro reverso a adicionar
    DN_novo_zona_reversa = "dlzZoneName=" + Dominio_reverso + ",ou=dns,dc=kirdeika,dc=org"
    DN_novo_reverso = "dlzHOSTName=" + Nome_curto_reverso + ",dlzZoneName=" + Dominio_reverso
+ ",ou=dns,dc=kirdeika,dc=org"
    DN_novo_reverso_soa = "dlzHOSTName=@,dlzZoneName=" + Dominio_reverso +
",ou=dns,dc=kirdeika,dc=org"
    DN_novo_reverso_soa_1 = "dlzRecordID=1,dlzHOSTName=@,dlzZoneName=" + Dominio_reverso +
",ou=dns,dc=kirdeika,dc=org"
    DN_novo_reverso_soa_2 = "dlzRecordID=2,dlzHOSTName=@,dlzZoneName=" + Dominio_reverso +
",ou=dns,dc=kirdeika,dc=org"
    DN_novo_reverso_soa_3 = "dlzRecordID=3,dlzHOSTName=@,dlzZoneName=" + Dominio_reverso +
",ou=dns,dc=kirdeika,dc=org"
    DN_novo_reverso_1 = "dlzRecordID=1,dlzHOSTName=" + Nome_curto_reverso + ",dlzZoneName=" +
Dominio_reverso + ",ou=dns,dc=kirdeika,dc=org"
    if Conexao_LDAP.add(DN_novo_direto, ['dlzHOST'], {'dlzHOSTName': Nome_curto}):
        total_registros += 1
    else:
        erros_LDAP += 1
    if Conexao_LDAP.add(DN_novo_direto_1, ['dlzARecord','top'], {'dlzHOSTName': Nome_curto,
'dlzIPAddr': IP, 'dlzRecordID': 1, 'dlzTTL': 3600, 'dlzType': 'a'}):
        total_registros += 1
    else:
        erros_LDAP += 1
    if Conexao_LDAP.add(DN_novo_zona_reversa, ['dlzZone','top'], {'dlzZoneName':
Dominio_reverso}):
        total_registros += 1
    if Conexao_LDAP.add(DN_novo_reverso_soa, ['dlzHOST'], {'dlzHOSTName': '@'}):
        total_registros += 1
    if Conexao_LDAP.add(DN_novo_reverso_soa_1, ['dlzSOARecord','top'], {'dlzAdminEmail':
'root.kirdeika.org.' ,'dlzExpire': 608400, 'dlzMinimum': 86400, 'dlzPrimaryNS':
'ns1.kirdeika.org.', 'dlzRefresh': 2800, 'dlzRetry': 7200, 'dlzSerial': 10, 'dlzHOSTName':
'@', 'dlzRecordID': 1, 'dlzTTL': 86400, 'dlzType': 'soa'}):
    if Conexao_LDAP.add(DN_novo_reverso_soa_2, ['dlzNSRecord','top'], {'dlzData':
'ns1.kirdeika.org.', 'dlzHOSTName': '@', 'dlzRecordID': 2, 'dlzTTL': 86400, 'dlzType': 'ns'}):
        total_registros += 1
    if Conexao_LDAP.add(DN_novo_reverso_soa_3, ['dlzNSRecord','top'], {'dlzData':
'ns2.kirdeika.org.', 'dlzHOSTName': '@', 'dlzRecordID': 3, 'dlzTTL': 86400, 'dlzType': 'ns'}):
        total_registros += 1
    if Conexao_LDAP.add(DN_novo_reverso, ['dlzHOST'], {'dlzHOSTName': Nome_curto_reverso}):
        total_registros += 1
    else:
        erros_LDAP += 1
    if Conexao_LDAP.add(DN_novo_reverso_1, ['dlzPTRRecord'], {'dlzData': Nome + ".",
'dlzHOSTName': Nome_curto_reverso, 'dlzRecordID': 1, 'dlzTTL': 3600, 'dlzType': 'ptr'}):
        total_registros += 1
    else:
        erros_LDAP += 1
def Conecta(HOST,Usuario,Senha,tls):
 global total_registros
 global erros_LDAP
 global erros_ssl
 try:
  s = Server(HOST=HOST, port=636, use_ssl=True, tls = tls)  # define an unsecure LDAP server,
requesting info on DSE and schema
  c = Connection(s, user=Usuario, password=Senha, auto_bind='NONE', version=3,
authentication='SIMPLE', client_strategy='SYNC', auto_referrals=True, check_names=True,
read_only=False, lazy=False, raise_exceptions=False)
  if not c.bind():
     exit(2)
  else:
     return c
 except LDAPException:
  erros_LDAP += 1
 except SSLError:
  erros_ssl += 1
def Desconecta(c):
 c.unbind()
```

```
# programa principal
if __name__ == "__main__":
 main()
```

Primeiro, é verificada a quantidade de entradas na base *LDAP:*

```
root@ca ~ # ldapsearch -D "cn=root,dc=kirdeika,dc=org" -w $senha -b
"ou=dns,dc=kirdeika,dc=org" | grep numEntries
# numEntries: 96
root@ca ~ # ldapsearch -D "cn=root,dc=kirdeika,dc=org" -w $senha -b "dc=kirdeika,dc=org" |
grep numEntries
# numEntries: 123
```

Constata-se um total de 123 entradas antes da execução do *script*. Em seguida, executa-se a inserção automática. As condições no *HOST* são de que nenhuma outra aplicação estivesse rodando que acesse o disco, de forma a realizar concorrência de acesso e consequente degradação de desempenho do *script* e da solução. Os dois servidores estão ativos, de forma que cada *insert* será replicado. Executa-se o *script.*

```
root@ca ~ # ./popula_dns.sh
Total de 259322 inserções LDAP
Erros LDAP                          = 8
Erros SSL                           = 0
Tempo total                         = 1:20:05.600140
```

O resultado informado pelo *script* leva a calcular que nos 4805 segundos de execução ocorreram 259322 inserções, o que leva a um desempenho de escrita de 53,9 inserções por segundo. A diferença entre o esperado e o aferido deve-se a erros de inserção, que não serão investigados, pois não há a necessidade devido ser uma inserção aleatória com o objetivo de obter um demonstrativo aproximado de desempenho.

Então, verifica-se novamente a quantidade de registros na base, acessando o servidor previamente configurado em /etc/openldap/ldap.conf da máquina ca, domain.kirdeika.org, cujo IP 192.168.0.1 está configurado pelo *UCARP* em serv_a.kirdeika.org:

```
root@ca ~ # ldapsearch -D "cn=root,dc=kirdeika,dc=org" -w $senha -b
"ou=dns,dc=kirdeika,dc=org" | grep numEntries
# numEntries: 259418
```

Para aferir o resultado, foram consultados os dois servidores separadamente.

```
root@ca ~ # ldapsearch -D "cn=root,dc=kirdeika,dc=org" -w $senha -b
"ou=dns,dc=kirdeika,dc=org" -H "ldaps://serv_b.kirdeika.org:636/" | grep numEntries
# numEntries: 259418

root@ca ~ # ldapsearch -D "cn=root,dc=kirdeika,dc=org" -w $senha -b
"ou=dns,dc=kirdeika,dc=org" -H "ldaps://serv_a.kirdeika.org:636/" | grep numEntries
# numEntries: 259418
root@ca ~ #
```

10.7 Teste de desempenho de consultas ao *OpenLDAP*

Para este teste foi desenvolvido outro *script*, em *python*, que usa o recurso de multiprocessamento para simular clientes simultâneos, realizando um determinado número de consultas.

Foi observado que o desempenho para o número inicial reduzido de consultas era extremamente baixo, duas consultas demoraram seis segundos, na mesma conexão. O análise é que como não havia um índice para o parâmetro dlzIPAddr, todos os registros da base de dados *LDAP* deveriam ser lidos para responder à requisição. Foi então criado um índice, usando a ferramenta *ldapmodify*, a partir de arquivo com o registro LDIF a seguir:

```
dn: olcDatabase={1}bdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: dlzIPAddr pres,eq
-
add: olcDbIndex
olcDbIndex: default pres,eq
```

Na versão do *LDAP* atual, com a configuração dinâmica, a criação de índices não exige a parada do serviço *LDAP*, como na versão anterior. Assim, o índice começou a ser criado ao término da execução do *ldapmodify*.

O novo resultado é que a consulta foi expressivamente mais rápida, de um total de seis segundos no primeiro teste, para 0,02 segundo após a criação do índice. Os comandos a seguir ilustram a situação.

```
mario@tatooine ~/pos_ceub/TCC/softwares/testes $ ./ldap-load.py 2 2
iniciando.... 2015-09-20 16:32:03.118627
iniciada thread Thread-1
1 threads fazendo 2 consultas = 2 requisições LDAP
Erros LDAP                    = 0
Erros ssl                     = 0
Tempo total                   = 0:00:06.300882

mario@tatooine ~/pos_ceub/TCC/softwares/testes $ ./ldap-load.py 2 2
iniciando.... 2015-09-20 16:37:55.786786
iniciada thread Thread-1
1 threads fazendo 2 consultas = 2 requisições LDAP
Erros LDAP                    = 0
Erros ssl                     = 0
Tempo total                   = 0:00:00.023972
```

Então, foram realizados outros testes com carga de conexões e consultas mais elevadas. Os testes foram realizados a partir da máquina ca, e os resultados podem ser observados na tabela 3.

Quadro 3 - Resultados de desempenho de consultas.

Teste	Threads	Consultas por thread	Total de consultas	Tempo decorrido em segundos	Desempenho em consultas por segundo	Erro
1	20	10	200	0,698032	286,5198	Não
2	20	100	2.000	2,996910	667,3540	Não
3	20	1.000	20.000	28,075657	712,3609	Não
4	100	10	1.000	2,390949	418,2437	Não
5	100	100	10.000	15,778254	633,7837	Não
6	100	1.000	100.000	145,048306	689,4255	Não
7	1.000	10	10.000	24,463812	408,7670	Não
8	1.000	100	100.000	160,355826	623,6131	Não
9	1.000	1.000	1.000.000	1.627,573517	614,4116	Não
Totais			1.243.200	2.007,381263	561,6088	Não

Fonte : Autor.

Pode-se então dizer que a média de desempenho de 1.243.200 consultas é de 561,6 consultas por segundo. Não foram objetivos do teste reconfigurações para desempenho expressivo, mas a alteração realizada propiciou, após a aferição do péssimo desempenho anterior de duas consultas a cada seis segundos, uma sensível melhora, o que reflete sensivelmente na consulta a registros DNS, principalmente.

Como sugestão de melhoria, deve-se aferir quais registros são mais importantes em pesquisas posteriores, a aplicações não constantes neste trabalho, de modo a propiciar desempenho otimizado à aplicação, criando-se os respectivos índices.

O *script* já descrito utilizado nos testes está a seguir.

```
#!/usr/bin/env python3
import random
import sys
import ssl
import os
import datetime
import threading
from LDAP3 import Server, Connection, LEVEL, BASE, SUBTREE, ALL, Tls, LDAPException
 # variáveis globais
errors_LDAP = 0
errors_ssl = 0
def main():
 # padrões deste script
 BaseDN = 'ou=dns,dc=kirdeika,dc=org'
 Usuario = 'cn=bind,dc=kirdeika,dc=org'
 HOST = 'domain.kirdeika.org'
 Senha = '$senha'
 CERTIFICADO_CA = '/root/rootCA.pem'
 # configurações TLS
 tls = Tls(validate=ssl.CERT_REQUIRED, version=ssl.PROTOCOL_TLSv1, ca_certs_file=CERTIFICADO_CA)
 n_consultas = int(sys.argv[2])
 tempo_inicio = datetime.datetime.now()
 print("iniciando.... %s" % (str(tempo_inicio)))
 total_threads = len(range(int(sys.argv[1])))
 print("repeat: %s " % (str(total_threads)))
 threads = []
 for repeat in range(total_threads):
  # conectando no OpenLDAP abrindo repeat theads fazendo 100 consultas
  nome_consulta = "consulta_" + str(repeat)
  t = threading.Thread(target=consultas, name=nome_consulta, args=(HOST,Usuario,Senha,tls,n_consultas))
  t.daemon = True
  threads.append(t)
  t.start()
  print("iniciada thread %s" % (str(t.getName())))
 for x in threads:
     x.join()
 tempo_fim = datetime.datetime.now()
 tempo_total = tempo_fim - tempo_inicio
 print("%s threads fazendo %s consultas = %s requisições LDAP" %
(str(total_threads),str(n_consultas),str(total_threads*n_consultas)))
 print("Erros LDAP                    = " + str(errors_LDAP))
 print("Erros ssl                     = " + str(errors_ssl))
 print("Tempo total                   = " + str(tempo_total))
 exit(0)
def Conecta(HOST,Usuario,Senha,tls):
 try:
  # define the server
  s = Server(HOST=HOST, port=636, use_ssl=True, tls = tls)  # define an secure LDAP server
  # define the connection
  c = Connection(s, user=Usuario, password=Senha, auto_bind='NONE', version=3, authentication='SIMPLE',
client_strategy='SYNC', auto_referrals=True, check_names=True, read_only=False,
               lazy = False, raise_exceptions = True, collect_usage = True)
  # perform the Bind operation
  return c
 except:
  print(c.result)
def Desconecta(c):
```

```
 c.unbind()
def consultas(HOST,Usuario,Senha,tls,n_consultas):
    global errors_LDAP
    global errors_ssl
    try:
     Conexao_1000 = Conecta(HOST,Usuario,Senha,tls)
     if not Conexao_1000.bind():
        print("conexao nao ok")
     else:
        for i in range(n_consultas):
          pesquisa(Conexao_1000)
     Desconecta(Conexao_1000)
    except LDAPException:
     errors_LDAP += 1
    except SSLError:
     error_ssl += 1
# função de pesquisa de registros, retorna lista de dn's dos achados por nome curto e IP
def pesquisa(Conexao):
 # define variaveis e listas
 Base = "ou=dns,dc=kirdeika,dc=org"
 Filter = "(dlzIPAddr=10." + str(random.randint(1,254)) + "." + str(random.randint(1,254)) + "." +
str(random.randint(1,254)) + ")"
 # pesquisa no LDAP por IP aleatório
 Conexao.search(search_base = Base,
        search_filter = Filter,
        search_scope = SUBTREE,
        attributes = ['dn'])
 result = Conexao.response
 lista_retorno = []
 for entry in result:
    DN = str(entry['dn'])
    lista_retorno.append(DN)
 return lista_retorno
# programa principal
if __name__ == '__main__':
 main()
```

CONCLUSÃO

O fácil acesso ao *software* livre é um incentivo a todos aqueles que desejam expandir seus conhecimentos. Mais que um incentivo, todo aquele que desejar usar um *software* livre, necessita realizar pesquisas não apenas na fonte do *software*, mas obter informações da maior quantidade possível de fontes de uso daquele *software*.

Foi experimentado, neste trabalho, a obtenção de informações de que os *softwares* utilizados eram usados sob as mais diversas formas, configurações e propósitos. Especificamente a forma de uso neste trabalho, foram poucas as abordagens encontradas, na forma de agregação proposta.

Assim, a todo o levantamento de informações, construção de configurações e testes realizados, atribui-se grande valor, pois agrega-se mais uma forma rica de uso ao *software* livre. Mais, atribui-se um valor agregado não apenas à construção realizada, mas a forma de se apresentar integrações de *softwares* que aparentemente não eram correlatos, que juntos entregam muito valor.

Com a pesquisa e construção realizadas, também conclui-se que uma grande necessidade de que quem deseja percorrer o novo caminho da Tecnologia da Informação dos próximos anos, deve tomar conhecimento não apenas da especificidade da área pela qual tem interesse, mas das áreas à sua volta.

Como exemplo, as novas tendências de entrega de valor agregado a infraestrutura levam a que os especialistas dessa área saibam que a construção de um servidor, nos moldes de antigamente, onde se propiciava todo o necessário ao *hardware*, e então aplicava-se uma ou outra tecnologia de *software* de modo a que o instalado entregasse algum serviço, mudou, exigindo que técnicas de desenvolvimento de *software* facilitem a automação dos processos de entrega de valor, no ponto em que aquela entrega feita em determinado tempo, que necessitava da intervenção de quase um dia, seja realizada em poucos minutos.

Observa-se a expansão de serviços em nuvem, notadamente *OpenSTACK*[15], e novas culturas como *DevOPS*[16]. Hoje, é exigido, e é uma situação rara, que um especialista de infraestrutura saiba programar. Isto foi necessário neste trabalho.

Ainda mais, o valor de apenas um serviço entregue é menor do que aquele com inteligência, onde a configuração de um *software* tem menor valor do que uma automação que use o *software* instalado como parte de um processo de entrega de um serviço de alto valor. Como construído neste trabalho, os dados do *OpenLDAP* tem pouco valor conparado ao seu uso em conjunto com os outros *softwares*, que usam estes dados para entregar um serviço de alto valor para uma ampla grade de usuários. A automação construída para o sistema tem tanto ou mais valor que a função de alta disponibilidade, com sua inteligência própria.

Foi de grande valor a aquisição de experiência no contato com novas linguagens de programação, exponencialmente a *python*. A forma de construção da linguagem e a documentação disponível, assim como vasta gama de exemplos de uso consoantes com as necessidades do trabalho facilitaram sobremaneira a sua consecução.

A metodologia proposta e as características intrínsecas ao *software* livre permitiram que se chegasse a um resultado utilizável em pouco tempo, com poucos recursos utilizados para a prova do conceito, e a visão de que a condensação dos conceitos e técnicas em apenas duas máquinas virtuais podem ser exponencialmente expandidas para uma solução que atenda a requisitos tanto de disponibilidade, abrangência e desempenho muito maiores, sem demasiado investimento de recursos.

---

15 - Conforme o site do projeto, *Openstack* é um sistema operacional em nuvem para controlar grande quantidade de recursos de computação, armazenamento e redes por *datacenters*, tudo controlado por uma interface que dá a administradores o controle de dar a seus usuários o poder de provisionar recursos por uma interface *web*. Disponível em <http://docs.openstack.org>. Acesso em 25 set. 2015

16 - Uma cultura que envolve a administração ágil de administração de sistemas, desenvolvimento e gerenciamento de projetos e suas associações, integrando as áreas antigamente conflitantes de desenvolvimento e infraestrutura. Nesta nova cultura, estas áreas são integradas e provém um ambientes de relacionamento saudável e ágil para o negócio, propiciando meios de responder a novas necessidades de mercado em termos de agilidade de entrega de produtos e serviços. Maiores informações em <http://devops.com>. Acesso em 25 set. 2015.

Os pontos chave do sucesso da pesquisa e implementação desta solução estão relacionados a conhecimentos de desenvolvimento de *software* entrelaçados a conhecimentos de infraestrutura. Esta relação é rara, por experiência do próprio autor em relação a seleção de pessoal para a área. Foi notada, durante os últimos anos, uma tendência dos profissionais de Tecnologia da Informação em especializar-se em sub-áreas, com detrimento de outras. Não tem sido interesse destes profissionais a obtenção de conhecimentos em áreas próximas e afins, e um interesse menor por infraestrutura. É difícil encontrar um profissional com o perfil e conhecimentos que permitam a sua plena evolução nesta área.

Este conhecimento do autor em relação a necessidades dos *softwares,* sob o ponto de vista do desenvolvimento de *software,* permitiu uma facilidade maior na construção das configurações dos diversos serviços, haja vista a necessidade de relacionamentos específicos entre eles. Não seria possível o desenvolvimento do *script* de atualização das informações do DNS a partir dos eventos DHCP, sem conhecimentos do *schema LDAP* envolvido na configuração da extensão *bind-dlz*, por exemplo. Há diversas destas interações neste trabalho, se não para dizer todas elas tem representações como esta.

Todos os objetivos propostos puderam ser alcançados sem grandes dificuldades, considerando a experiência do autor com a automação dos processos e inteligência embutida em programação para administração de servidores.

A maior dificuldade de uso do *software* livre foi experimentada pelo autor na consecusão dos objetivos deste trabalho, que é a profusão de possibilidades de uso inerente aos *softwares* livres escolhidos, que leva a uma necessidade anterior de definição de requisitos, premissas e possibilidades.

Em caso da expansão do projeto com a necessidade de contratação, é primordial o conhecimento em relação a *software* livre, desenvolvimento e infraestrutura nos profissionais envolvidos, e a seleção destes profissionais atendendo a estes requisitos. Esta seleção envolve enorme dificuldade pelos motivos apontados com relação ao desenvolvimento profissional comumente

verificado.

Também, a documentação não completa dos *softwares* foi um complicador, daí a necessidade de maior pesquisa a respeito de usos dos *softwares,* possibilidades e exemplos. Pontualmente, isso foi experimentado na extensão *bind-dlz* e no uso do *LDAP* com DHCP.

O perfil do profissional de *software* livre diz sobre uma pessoa investigativa, curiosa e insatisfeita com seus próprios conhecimentos, e esta situação demonstra uma pessoa que constantemente pesquisa, e que portanto tem grandes possibilidades de entrar em contato com as mais recentes tecnologias. Daí a maior dificuldade, tendo em vista a tendência à acomodação, ao costume à zona de conforto da maioria dos profissionais da área. Poucos saem ou estão dispostos a saírem desta zona de conforto.

Assim, a pesquisa, estudo, testes e experimentações, a aceitação de que algo ou a forma realizada não funciona e deve ser mudado, foram as situações enfrentadas neste trabalho.

Mas a realização plena de todos os objetivos, e a percepção de que muito mais poderia ser agregado, na forma de aplicações da estrutura, evolução e introdução de novos conceitos e tecnologias agregando valor a esta proposta, tornou a execução de trabalho de grande satisfação.

Foi notado que a inclusão de algumas políticas marginais à solução seriam necessárias à aplicação da proposta em determinadas situações. Por exemplo, o monitoramento da replicação do *OpenLDAP*, em caso de mais nós, ou em caso dos servidores estarem interligados por uma estrutura física que possa apresentar uma instabilidade maior que a experimentada neste trabalho pode se tornar uma premissa, com avisos ao administrador de modo a manter a confiabilidade na solução, do ponto de vista da integridade e disponibilidade da informação.

Igualmente, a evolução dos *scripts* de testes dos serviços, de modo a dar mais consistência nos avisos em caso de problemas, seria o próximo passo evolutivo. A mesma visão pode ser aplicada ao *UCARP*.

A integração aqui proposta pode servir de base ou inspiração para outras integrações de tecnologias, em que a interação de serviços aparentemente isolados podem fornecer soluções para grandes problemas se funcionarem juntos.

São sugestões para próximos trabalhos a expansão do proposto aqui com relação ao já mencionado, e ainda a melhoria da disponibilidade e desempenho dos sistemas de DNS, expandido o ISC-BIND para uma visão de *Master-Slave*, com os *Masters* apresentados na forma aqui descrita, na expansão dos servidores DHCP locais para outras redes, usando a base de dados de *OpenLDAP* remotamente, centralizando as configurações, ou até mesmo expandido o aqui proposto agregando funções como roteamento de rede, *firewalls* internos, externos e de aplicação, e a expansão do projeto proposto para trabalhar com IPv6.

REFERÊNCIAS

ADRIAN, David et al. **Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice**, 2015. Disponível em: <https://weakdh.org/> . Acesso em: 02 jun. 2015.

ARCHLINUX.ORG. **Apache HTTP Server**. 2015. Disponível em: <https://Wiki.ArchLinux.org/index.php/Apache_HTTP_Server>. Acesso em: 04 ago. 2015.

ARCHLINUX.ORG. **Arch Build System**. 2015. Disponível em: <https://Wiki.ArchLinux.org/index.php/Arch_Build_System>. Acesso em: 04 ago. 2015.

ARCHLINUX.ORG. **Dhcpd**. 2015. Disponível em: <https://Wiki.ArchLinux.org/index.php/Dhcpd>. Acesso: em 05 ago. 2015.

ARCHLINUX.ORG. **Iptables.** 2015. Disponível em: <https://Wiki.ArchLinux.org/index.php/Iptables>. Acesso: em 22 jul. 2015.

ARCHLINUX.ORG. **LDAP_authentication**. 2015. Disponível em: <https://Wiki.ArchLinux.org/index.php/LDAP_authentication>. Acesso em: 23 jul. 2015.

ARCHLINUX.ORG. **LVM**. 2015. Disponível em: <https://Wiki.ArchLinux.org/index.php/LVM>. Acesso em: 23 jul. 2015.

ARCHLINUX.ORG. **Network Configuration**. 2015. Disponível em: <https://Wiki.ArchLinux.org/index.php/Network_configuration>. Acesso em: 22 jul. 2015.

ARCHLINUX.ORG. **OpenLDAP**. 2015. Disponível em: <https://Wiki.ArchLinux.org/index.php/OpenLDAP>. Acesso em: 23 jul. 2015.

ARCHLINUX.ORG. **Partitioning**. 2015. Disponível em: <https://Wiki.ArchLinux.org/index.php/Partitioning>. Acesso em: 23 jul. 2015.

ARCHLINUX.ORG. **Syslog-ng**. 2015. Disponível em: <https://Wiki.ArchLinux.org/index.php/Syslog-ng>. Acesso em: 23 jul. 2015.

ARCHLINUX.ORG. **VirtualBox**. 2015. Disponível em: <https://Wiki.ArchLinux.org/index.php/VirtualBox>. Acesso em: 21 jul. 2015.

CASSEN, Alexandre. **KeepAlived**, 2015. Disponível em: <http://www.keepalived.org>, Acesso em: 01 mar. 2015.

DREYFUS. Emmanuel. **TLS hardening**. 2014. Disponível em <http://bsdmag.org/tls/>. Acesso em: 29 jul. 2015.

FREEBSD. **Common Address Redundancy Protocol (CARP)**, 2015. Disponível em: <https://www.freebsd.org/doc/en/books/handbook/CARP.html> , Acesso em: 25 mai. 2015

GIL, Anahuac de Paula. **OpenLDAP Extreme**. Rio de Janeiro : BRASPORT, 2012.

IETF. RFC 1035 – **DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION**. Network Working Group, 1997. Disponível em: <https://www.ietf.org/rfc/rfc1035.txt>. Acesso em: 28 fev. 2015.

IETF. RFC 2131 – **Dynamic HOST Configuration Protocol. Network Working Group**, 1997. Disponível em: <https://www.ietf.org/rfc/rfc2131.txt>. Acesso em: 28 fev. 2015.

IETF. RFC 2338 – **Virtual Router Redundancy Protocol**, 1998. Disponível em: <https://www.ietf.org/rfc/rfc2131.txt>. Acesso em: 25 mai. 2015.

IETF. RFC 3074 – **DHCP Load Balancing Algorithm**. Network Working Group, 2001. Disponível em: <https://www.ietf.org/rfc/rfc3074.txt>. Acesso em: 28 fev. 2015.

IETF. RFC 4511 – **Lightweight Directory Access Protocol (LDAP): The Protocol**. Network Working Group, 2006. Disponível em: <https://www.ietf.org/rfc/rfc4511.txt>. Acesso em: 28 fev. 2015.

IETF. RFC 5246 – **The Transport Layer Security (TLS) Protocol Version 1.2**. Network Working Group, 2008. Disponível em: <https://www.ietf.org/rfc/rfc5246.txt>. Acesso em: 19 jan. 2016.

IETF. RFC 7568 – **Deprecating Secure Sockets Layer Version 3.0**. Network Working Group, 2015. Disponível em: <https://www.ietf.org/rfc/rfc7568.txt>. Acesso em: 19 jan. 2016.

INTERNET SYSTEMS CONSORTIUM. **BIND9**. 2015. Disponível em: </https://www.isc.org/downloads/bind>. Acesso em: 02 mar. 2015.

INTERNET SYSTEMS CONSORTIUM. **DHCP**. 2015. Disponível em: <https://www.isc.org/downloads/DHCP>. Acesso em: 02 mar. 2015.

MENAGE, Paul. **CGROUPS.** Disponível em: <https://www.kernel.org/doc/Documentation/CGROUPS/CGROUPS.txt>. Acesso em: 27 jul. 2015.

MOORE, Don. **DNS server survey.** 2004. Disponível em: <http://mydns.bboy.net/survey/>. Acesso em: 02 mar. 2015.

OPENLDAP FOUNDATION. **OpenLDAP**. 2015. Disponível em: <http://www.OpenLDAP.org>. Acesso em: 02 mar. 2015.

OPENLDAP FOUNDATION. **Replication**. 2015. Disponível em:

<http://www.OpenLDAP.org/doc/admin24/replication.html>. Acesso em: 03 jul. 2015.

OPENSSL PROJECT. **OpenSSL**. 2014. Disponível em: <https://www.OpenSSL.org/>. Acesso em: 29 jul. 2015.

OWASP. **Transport Layer Protection Cheat Sheet.** 2015. Disponível em: <https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet> . Acesso em: 29 jul. 2015.

RASPET, Sunny. **CARP your way to high availability**. 2004. Disponível em: <http://archive09.Linux.com/feature/35482>. Acesso em: 25 mai. 2015.

RISTIĆ, Ivan. **SSL/TLS Deployment Best Practices**, 2014. Disponível em: <https://www.ssllabs.com/downloads/SSL_TLS_Deployment_Best_Practices.pdf> . Acesso em: 02 jun. 2015.

VINET, Judd. GRIFFIN, Aaron. **ArchLinux.ORG**. 2015. Disponível em: <https://www.ArchLinux.org> . Acesso em: 02 mar. 2015.

WEBER, Taisy Silva. **Tolerância a falhas: conceitos e exemplos**. Apostila do Programa de Pós-Graduação–Instituto de Informática-UFRGS. Porto Alegre, 2003.

WEBER, Taisy Silva. **Um roteiro para exploração dos conceitos básicos de tolerância a falhas**. Porto Alegre, 2002. Disponível em: <www.inf.ufrgs.br/~taisy/disciplinas/textos/Dependabilidade.pdf>. Acesso: em 01 mar. 2015.

APÊNDICE A – CRIAÇÃO DE MÁQUINA VIRTUAL NO *VirtualBox*

A imagem para a instalação destas máquinas pode ser copiada a partir do *site* do *ArchLinux*, disponível em <https://www.*archlinux*.org/download/> , onde há explicações sobre os diversos métodos de *download*. No momento da escrita deste trabalho, a versão 2015.07.01, de tamanho 646MB, com o *kernel* 4.0.7 estava disponível. Trata-se de imagem composta pelas versões 32 e 64 bits do sistema. É essencial a verificação da soma SHA1 ou MD5 da imagem copiada com a soma disponível no *site*, para confirmação da sua integridade.

A criação de uma máquina virtual no *VirtualBox* se inicia clicando-se em novo na interface. Na janela Criar máquina Virtual, dá-se um nome à *vm*, seleciona-se um tipo e versão, como na figura 4.

Clicando-se em próximo, altera-se o tamanho da memória. Neste momento, 1024MB são suficientes, e serão ajustados conforme a necessidade da máquina virtual. Destaca-se aqui que em se tratando de máquinas virtuais, não é boa prática, como é costume em máquinas físicas, ajustar-se a memória *swap* a duas vezes a memória RAM, pois o acesso a disco também é compartilhado entre as diversas *vm's* instaladas, podendo causar um colapso no acesso ao disco. Então, o ideal é que tenha disponível toda a memória RAM necessária às aplicações, e monitorando-se o uso de *swap*, que é mantido em níveis razoavelmente míninos, ajusta-se esta quantidade de memória RAM para que contenha as aplicações.

Clicando-se em próximo, configura-se a criação de disco rígido. Neste momento vale ressaltar que para a aplicação proposta, não é necessária grande quantidade de disco, pois não será armazenada grande quantidade de informação. Pode-se adicionar espaço posteriormente, em caso de necessidade, tanto criando um novo disco, maior, e clonando-se o disco antigo sobre o novo, quanto adicionando-se discos e adicionando-os ao volume lógico associado, usando-se discos LVM.

Será usado um disco de 20GB, com alocação dinâmica. Para se criar um disco, nesta tela seleciona-se Criar um disco virtual agora, conforme a figura 5, e em seguida clica-se em criar. Na tela seguinte, conforme a figura 6, seleciona-se o tipo deste disco, recomendado o tipo nativo do *VirtualBox*, VDI. Clicando-se em próximo seleciona-se o tipo de armazenamento, para Dinamicamente alocado, conforme a figura 7. Clicando-se em próximo escolhe-se o local no *HOST* do armazenamento do disco virtual e seu tamanho, conforme a figura 8. É boa prática dar-se ao disco virtual o mesmo nome da *vm*, por facilidade.

Figura 5 - Criação de máquina virtual, seleção de tipo, versão e nome da vm.

Fonte: Autor.

Figura 6 - Criação de disco Virtual.

Fonte: Autor.

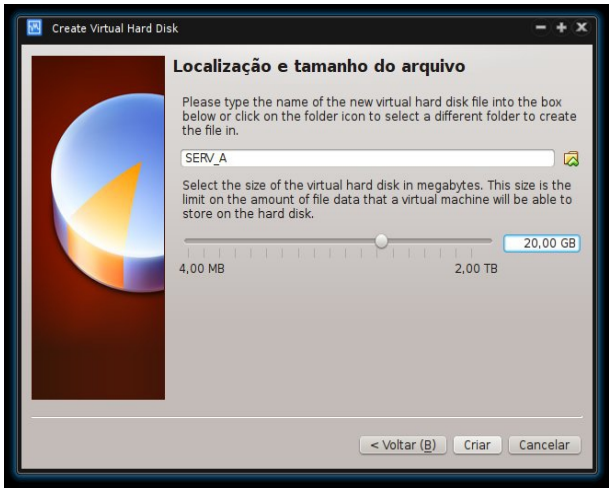
Figura 7 - Tipo de disco a ser criado.

Fonte: Autor.

Figura 8 - Tipo de alocação do disco.

Fonte: Autor.

Figura 9 - Tamanho do disco a ser criado.

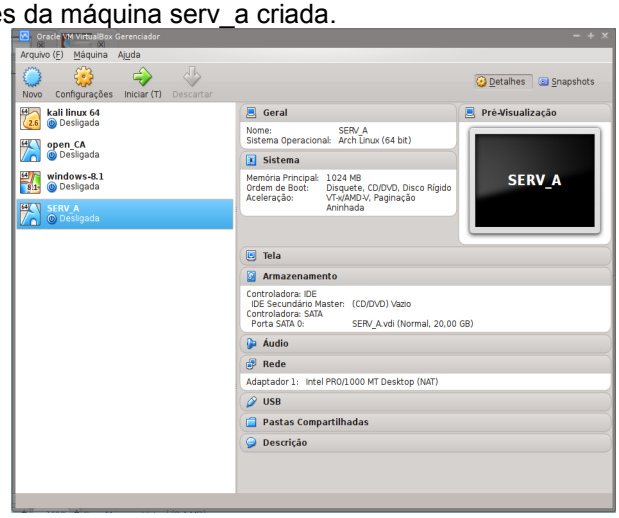


Fonte: Autor.

O local padrão para armazenamento dos discos é definido em Arquivo → Preferências → Geral → Pasta Padrão para máquinas , ou escolhido clicando-se no símbolo de pasta ao lado do campo do nome. Finaliza-se clicando em criar. Neste momento a máquina virtual está criada, mas é necessário a configuração de rede e outros parâmetros.

Neste momento, a máquina serv_a destacada foi criada, e tem as configurações informadas conforme a figura 10.

Figura 10 - Configurações da máquina serv_a criada.



Fonte: Autor.

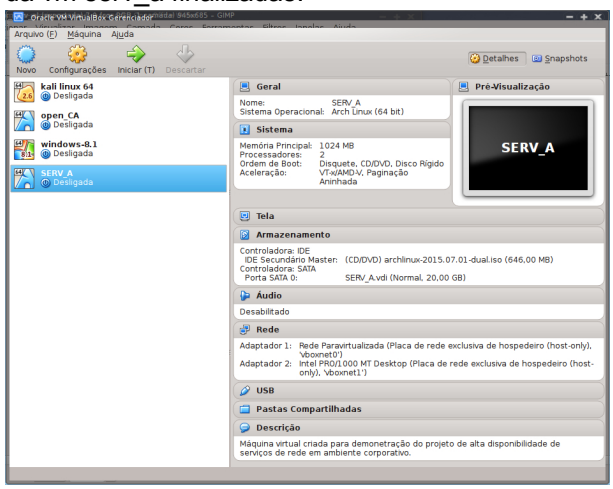
Para as necessidades deste projeto, clica-se em Configurações. Em Geral, na aba Avançado, altera-se a Área de transferência compartilhada e Arrastar e soltar para Bi-direcional. Na aba Descrição, é conveniente informar dados sobre a *vm* criada, para documentação. Em Sistema (à direita), na aba Placa-Mãe, seleciona-se o *chipset* ICH9.

Na aba Processador, convém associar dois processadores, mas isso será de acordo com as capacidades do *HOST*, balanceando as necessidades de processamento, capacidade do *HOST* e vantagens do uso de multiprocessamento do sistema e aplicações da *vm*.

Em Armazenamento, na Controladora SATA, ativa-se o Utilizar *cache* de I/O do hospedeiro. Na Controladora IDE, pode-se neste momento associar a imagem baixada do *ArchLinux* para *boot* e instalação, clicando-se no símbolo de CD à direita na seção Atributos. Em Áudio, pode-se desabilitar o áudio, pois não será usado. Em Rede, na aba Adaptador 1, habilita-se a placa, seleciona-se Conectado a para Placa de rede exclusiva do hospedeiro, seleciona a rede *vboxnet0* em Nome, e em Avançado seleciona-se o Tipo de placa para Rede Paravirtualizada e em Modo Promíscuo seleciona-se Permitir Tudo. Ativa-se o Cabo conectado. Na aba Adaptador 2, habilita-se a placa, seleciona-se Conectado a para Placa de rede exclusiva do hospedeiro, seleciona a rede *vboxnet1* em Nome, e em Avançado seleciona-se o Tipo de placa para Rede Paravirtualizada e em Modo Promíscuo seleciona-se Permitir Tudo. Ativa-se o Cabo conectado e pode-se clicar em OK na parte inferior da tela, finalizando-se a configuração.

A figura 11 mostra as alterações realizadas na configuração da *vm*.

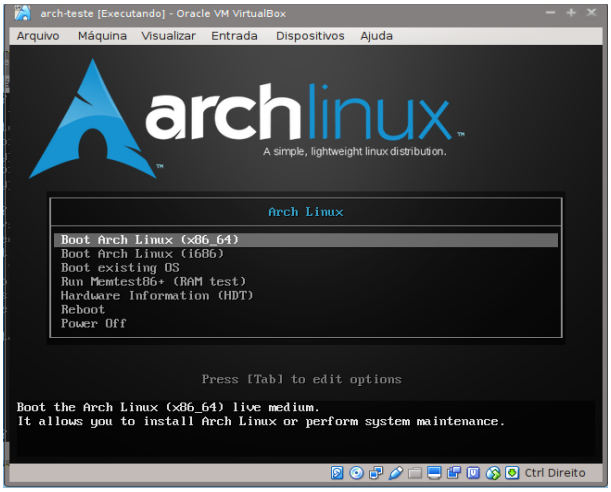
Figura 11 - Configurações da vm serv_a finalizadas.



Fonte: Autor.

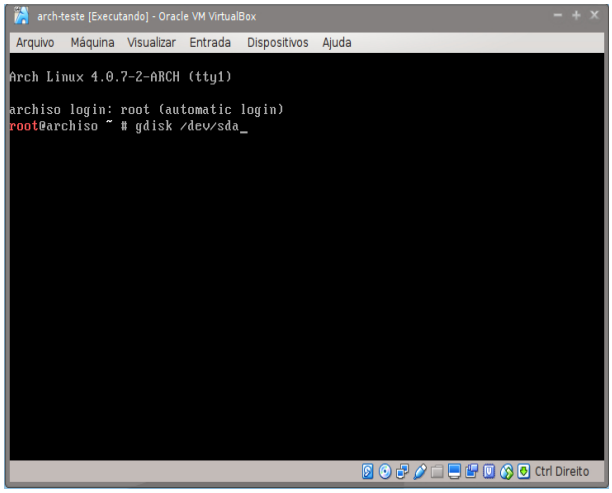
Neste momento pode-se iniciar a máquina virtual e iniciar a instalação do sistema operacional, como na figura a seguir.

Figura 12 - *Boot* da máquina virtual com a imagem.



Fonte: Autor.

Figura 13 - *Boot* do *ArchLinux* concluído.



Fonte: Autor.

APÊNDICE B – INSTALAÇÃO DO *ARCHLINUX* EM MÁQUINA VIRTUAL

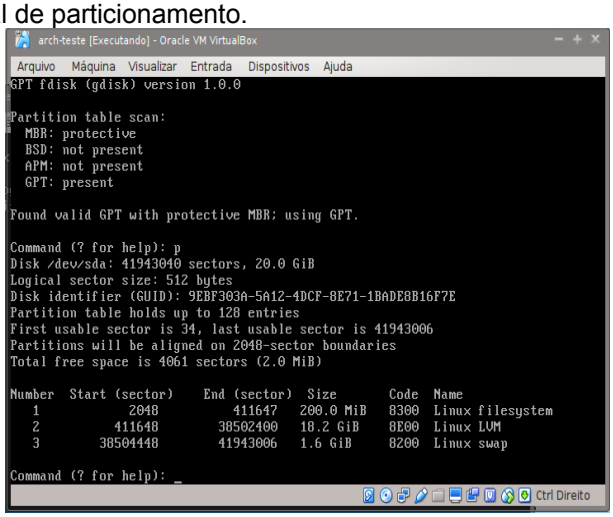
Para efeitos de instalação do sistema, será assumido o seguinte esquema de particionamento de disco:

```
/     =     2,8GB
/usr  =     8GB
/var  =     8GB
/boot =     200MB
swap  =     1GB
```

Sendo que cada diretório deste esquema será um dispositivo LVM, a exceção do /boot e swap. Seguindo as recomendações do manual de particionamento do *ArchLinux*(2015), a respeito da escolha entre os tipos de tabelas de partição GPT e MBR, é escolha a GPT por ser compatível com o *VirtualBox*, permitir mais partições que as limitadas 4 do MBR, permitir *boot* em *firmwares* EFI onde aplicável, e ter sido desenvolvido como sucessor do arcaico MBR, que remonta à época do MS-DOS. Após o *boot* da máquina virtual com a imagem de instalação anexada do *drive* de *cd*, usando a imagem de 64bits, iniciamos o particionamento do disco usando o utilitário gdisk, passando como parâmetro o disco.

Em seguida, criamos uma tabela de partições tipo GPT digitando “o” e “Y”. Após, uma nova partição é adicionada digitando “n” , “1”, obedecemos ao primeiro cilindro informado. Para esta primeira partição, usaremos como próximo parâmetro o tamanho, que pertencerá ao “/boot” com 200MB, digitando “+200M”. O tipo de sistema de arquivo é o padrão, “*Linux* “filesystem”. Para os pontos de montagem restantes, é essencial o uso de LVM, para flexibilidade posterior na expansão do espaço em disco, se necessário, assim, a próxima partição terá 18,8GB, soma das restantes, do tipo “*Linux* LVM”, e dentro deste dispositivo LVM estes espaços serão criados. Para o *swap*, o tipo de sistema de arquivos deverá ser mudado para 8200, “*Linux* swap”. O comando “p” mostra a tabela de partições criada. A figura 13 ilustra esta tabela. Grava-se com o comando “w” e saímos do programa com “q”.

Figura 14 - Esquema final de particionamento.



Fonte: Autor.

Terminado o particionamento, deve-se criar o grupo físico LVM, o grupo de volumes lógicos e os volumes lógicos, formatar as partições, criar os pontos de montagem e efetivamente montar o sistema em /mnt, conforme solicita o manual de instalação do *ArchLinux*. Os comandos a seguir , conforme a seção LVM do *Wiki* do *ArchLinux* (2015), realizam estas tarefas:

```
# vgcreate VolGroupserv_a /dev/sda2                    - cria o pv e o vg
# lvcreate -L 2800M -n lv_root VolGroupserv_a          - lv “/”
# lvcreate -L 8G -n lv_usr VolGroupserv_a              - lv “/usr”
# lvcreate -l +100%FREE -n lv_var VolGroupserv_a       - lv “/var”
# mkfs.ext2 /dev/sda1                                  - “/boot”
# mkfs.ext4 /dev/mapper/VolGroupserv_a-lv_root         - fs “/”
# mkfs.ext4 /dev/mapper/VolGroupserv_a-lv_usr          - fs “/usr”
# mkfs.ext4 /dev/mapper/VolGroupserv_a-lv_var          - fs “/var”
# mkswap /dev/sda3                                     - fs swap
# mkdir -p /mnt/{boot,usr,var}                         - pontos de montagem
# mount -t ext4 /dev/mapper/VolGroupserv_a-lv_root /mnt
# mount -t ext4 /dev/mapper/VolGroupserv_a-lv_usr /mnt/usr
# mount -t ext4 /dev/mapper/VolGroupserv_a-lv_var /mnt/var
# mount -t ext2 /dev/sda1 /mnt/boot
# swapon /dev/sda3                                     - ativando swap
```

Com isto, a criação do esquema de particionamento de disco está concluída e disponível para o restante do processo de instalação do sistema

operacional. A instalação do *ArchLinux* depende de acesso à *Internet*, portanto a rede deve ser configurada para permitir este acesso. Os comandos a seguir realizam a tarefa, tendo como base os endereços que serão usados no projeto:

```
# ip addr add 192.168.0.10/24 dev eth0
# ip route add default via 192.168.0.100 dev eth0
# echo “nameserver 192.168.0.100” >> /etc/resolv.conf
```

Com isso, podem ser selecionados os espelhos a partir dos quais a instalação será realizada. Os locais de espelho dos pacotes da distribuição são configuráveis em /etc/pacman.d/mirrorlist. No momento da escrita deste trabalho, foi constatado como o mais rápido o espelho instalado na Universidade de São Paulo (USP), disponível em <http://linorg.usp.br/archlinux/$repo/os/$arch>.

O comando a seguir, de uma única linha, executa esta configuração sem a necessidade de editor de texto.

```
# sed -i '1 i Server = http://linorg.usp.br/archlinux/$repo/os/$arch'
/etc/pacman.d/mirrorlist
```

Após, usando o *script pacstrap*, instala-se o sistema básico. Gera-se então o /etc/fstab a partir da construção realizada com o *genfstab*. Muda-se a raiz do sistema corrente para a nova instalação com o *arch-chroot*. Define-se o nome da máquina, arquivo /etc/hosts, o *Timezone* e as configurações de *locales*, gerando-se em seguida estes arquivos de *locale* e definindo a linguagem do sistema. São os comandos a seguir:

```
# pacstrap /mnt base
# genfstab -p /mnt  >> /mnt/etc/fstab
# arch-chroot /mnt
# echo “serv_a.kirdeika.org” >> /etc/hostname
# echo “192.168.0.10 serv_a serv_a.kirdeika.org” >> /etc/HOSTs
# ln -sf /usr/share/zoneinfo/America/Sao_Paulo /etc/localtime
# sed -i '/pt_BR/s/^#//' /etc/locale.gen
# sed -i '/en_US/s/^#//' /etc/locale.gen
# locale-gen
# echo LANG=pt_BR.UTF-8 > /etc/locale.conf
```

Foi constatado um problema com a composição de teclas usando o equipamento *HOST* descrito. No seu teclado, os caracteres “/” e “?” são compostos com o “AltGr” seguido dos caracteres “q” e “w”, situação não prevista pelo arquivo normalmente usado /usr/share/kbd/keymaps/i386/qwerty/br-abnt2.map.gz. Assim, um novo arquivo foi construído a partir deste, adicionando-se as linhas “altgr keycode 16 = slash” e “altgr keycode 17 = question” ao arquivo acima descompactado. O mesmo foi novamente compactado como /usr/share/kbd/keymaps/i386/qwerty/br-abnt2-note.map.gz e carregado, habilitando as teclas não funcionais. Sendo assim, o arquivo /etc/vconsole.conf ficou da forma a seguir:

```
UNICODE="1"
KEYMAP="br-abnt2-note"
KEYMAP_CORRECTIONS="euro2"
FONT="lat9u-16"
LEGACY_CHARSET="iso-8859-15"
BROKEN_COMPOSE="0"
```

Conforme a seção *Mkinitcpio* do *Wiki* do *ArchLinux*(2015), quando o diretório “/usr” é usado em partição separada, configurações adicionais devem ser realizadas. O serviço *mkinitcpio-generate-shutdown-ramfs.service* deve ser habilitado, ou o “hook” “shutdown” deve ser adicionado à seção “HOOKS” de “/etc/mkinitcpio.conf”. A segunda opção foi escolhida neste projeto pois já poderia ser habilitada neste momento, ao contrário a nova máquina não completaria o *boot* pois diversos executáveis importantes estão presentes naquele diretório. Ainda, conforme o *Wiki*, o “hook” “fsck” deve ser adicionado e o “/etc/fstab” deve ter a seção “passno” definida como “0” para /usr. Esta situação força a verificação deste sistema de arquivos no *boot.* Ainda, como se está usando sistemas de arquivos em LVM, o “hook” “lvm2” igualmente deve ser adicionado neste arquivo. Realizadas estas configurações, cria-se o RAM *disk* inicial com o comando a seguir e então define-se a senha do superusuário com *passwd*. Pode-se então sair do ambiente chroot com “CTRL+D” após o comando:

```
# mkinitcpio -p Linux
```

Para que o sistema possa ser carregado, um gerenciador de

inicialização deve ser instalado. O padrão do sistema é o *grub2*, havendo diversas outros opções, conforme a seção *Grub* do *Wiki* do *ArchLinux*(2015). Para a instalação realizada, segue-se com a criação de uma partição do tipo BIOS *boot*, com o *gdisk*. Cria-se uma nova partição no disco /dev/sda, não importando a ordem em relação às outras partições, iniciando-se no setor 34 até o setor 2047, do tipo “ef02”, “BIOS boot”. Grava-se com “w” e “q” para sair do programa.

Deve-se então voltar ao ambiente *chroot*, com “arch-chroot /mnt”. Instala-se o gerenciador *grub* com “pacman -S grub”. Em seguida instala-se o gerenciador em disco com os comandos:

```
# grub-install --target=i386-pc --recheck --debug /dev/sda
# grub-mkconfig -o /boot/grub/grub.cfg
```

Os avisos do LVM podem ser ignorados. Finalizados estes procedimentos, o sistema está então instalado. Deve-se em seguida sair do ambiente *chroot* com “CTRL+D”, e desligar a máquina virtual com o comando “poweroff”. Então desliga-se a conexão da imagem de instalação do drive emulado de *cdrom* da *vm* no *VirtualBox*, podendo-se então iniciá-la novamente, totalmente no ambiente instalado.

Segue-se configurando a rede, a partir das instruções da seção *systemd-networkd* do *Wiki* do *ArchLinux*(2015). São habilitados e iniciados os serviços *systemd-networkd* e *systemd-resolved*, e criado um *link* do arquivo /etc/resolv.conf criado pelo sistema *systemd* ao arquivo padrão do sistema, com os comandos a seguir:

```
# systemctl enable systemd-networkd
# systemctl enable systemd-resolved
# systemctl start systemd-networkd
# systemctl start systemd-resolved
# rm -f /etc/resolv.conf
# ln -s /run/systemd/resolve/resolv.conf /etc/resolv.conf
```

É necessária a criação de dois arquivos de configuração para a rede, descritos a seguir:

```
### conteúdo de /etc/systemd/network/eth0.network ===
[Match]
Name=eth0

[Network]
DNS=192.168.0.100
Address=192.168.0.10/24
Gateway=192.168.0.100

### conteúdo de /etc/systemd/network/eth1.network ===
[Match]
Name=eth1

[Network]
Address=172.16.0.10/24
```

Então, reinicia-se os serviços *systemd-networkd* e *systemd-resolved*, validando as alterações e ativando a rede.

```
# systemctl restart systemd-networkd
# systemctl restart systemd-resolved
```

Cria-se neste momento um usuário comum, para realização de acessos ao sistema e tarefas. Após a inserção da senha inicial do usuário, pode-se realizar o *login* deste usuário com su. Usa-se o comando a seguir para adicionar um usuário.

```
# useradd -d /home/mario -m -U -s /bin/bash mario
```

Para a instalação de *CGROUPS*, dois pacotes básicos devem ser instalados. Do repositório principal, o pacote *cgmanager*. Porém, as ferramentas em espaço de usuário *libcgroup* estão apenas disponíveis no repositório de responsabilidade de usuário AUR (*Arch User Repository*).

Ferramentas para manipulação de dependências de forma automatizada também estão disponíveis para interação com este repositório, porém alguns pacotes iniciais devem ser instalados, como o grupo básico para desenvolvimento.

Os pacotes do AUR consistem em arquivos contendo instruções para construção dos pacotes para instalação em sistemas *ArchLinux*, cujo código fonte

original é essencialmente copiado do local do projeto disponibilizado pelos respectivos autores, e minimamente alterados para uso no *ArchLinux*. Os comandos a seguir instalam o sistema básico para compilação, baixam a ferramenta *packer-color* e instalam os pacotes necessários ao *CGROUPS*:

```
# pacman -Sy –needed base-devel jshon git wget
# echo “mario ALL=(ALL) PASSWD: ALL” >> /etc/sudoers
# su – mario
$ wget https://aur4.archlinux.org/cgit/aur.git/snapshot/packer-color.tar.gz
$ tar xzvf packer-color
$ cd packer-color
$ makepkg
$ sudo pacman -U packer-color-20130730-1-any.pkg.tar.xz
$ packer-color -s cgmanager libcgroup
$ exit
```

Habilita-se e inicia-se o serviço com os comandos a seguir.

```
# systemctl enable cgmanager
# systemctl start cgmanager
```

Prossegue-se com a instalação do *VirtualBox Guest Aditions*, que pode ser instalado com o pacote *VirtualBox-guest-utils*. Com isso, drivers e aplicações que otimizam o ambiente virtual são instalados. Então, deve ser habilitado e iniciado o serviço *vboxservice* que carrega os módulos do *kernel vboxguest*, *vboxsf* e *vboxvideo*, otimizando a máquina virtual. Os comandos a seguir realizam as tarefas.

```
# pacman -Sy VirtualBox-guest-utils
# systemctl enable vboxservice
# systemctl start vboxservice
```

Com estes procedimentos concluídos, o sistema operacional da máquina serv_a está instalado.

APÊNDICE C – *SCRIPTS* DA SOLUÇÃO DE ALTA DISPONIBILIDADE

1 Update_DNS

O algoritmo do *script* encontra-se descrito nas figuras 15 a 19 nas páginas a seguir, e parte do princípio de que o servidor DHCP pode enviar as informações necessárias à edição da base de dados do DNS, de forma a refletir o endereço de rede referente a uma solicitação atendida.

É utilizada uma nova funcionalidade do servidor ISC-DHCP, em que a cada *lease*, ou seja, um evento de fornecimento de endereço a um cliente, um programa pode ser executado, com uma inteligência de programação interna associada ao evento. Com esta facilidade, foram construídas as regras constantes como *on_commit*, *on_release* e *on_expiry* nas linhas a seguir, parte da configuração presente na árvore *LDAP* correspondente ao servidor DHCP.

```
dhcpStatements: on commit {      set ClientName = concat("dhcp-", binary-to-as
 cii(10, 8, "-", leased-address), ".", config-option domain-name);     set Cli
 entIP = binary-to-ascii(10, 8, ".", leased-address);       set ClientPTR = co
 ncat (binary-to-ascii (10, 8, ".", reverse (1, leased-address)),".IN-ADDR.ARP
 A");      execute("/etc/openldap/update_dns.sh", "add", ClientIP, ClientPTR,
 ClientName);  }
dhcpStatements: on expiry  {set ClientIP = binary-to-ascii(10, 8, ".", lease
 d-address) ;  execute ("/etc/openldap/update_dns.sh", "delete", ClientIP);
  }
dhcpStatements: on release {set ClientIP = binary-to-ascii(10, 8, ".", lease
 d-address) ;  execute ("/etc/openldap/update_dns.sh", "delete", ClientIP);
  }
```

Conforme pode-se observar, são construídas dinamicamente as informações necessárias à atualização do serviço DNS, através de inserção / remoção direta dos dados na base *LDAP*, servindo-se amplamente da funcionalidade de carregamento dinâmico de zonas do ISC-BIND.

Observação importante é que se o cliente estiver previamente cadastrado no serviço DHCP, e o endereço cadastrado corresponda ao endereço físico de rede (MAC *address*) de origem da solicitação, não será gerado um evento de *release* ou *expiry* e portanto não serão removidos os registros do DNS quando da

ocorrência destes eventos, ou seja, quando o cliente informar a devolução do endereço fornecido, ou quando este fornecimento expirar, respectivamente.

Assim, incluir um novo registro, remover ou alterar um existente são ações necessárias a este *script*. Foram considerados os seguintes requisitos, de conformidade com o *schema* do DNS presente no *OpenLDAP* e premissas do projeto:

- o *script* deve ser executado localmente;
- o *script* deve poder atualizar um registro existente, mantendo a informação sobre o nome de máquina e endereço consistentes;
- o *script* deve remover completamente o registro quando solicitado;
- o *script* deve permitir a execução de várias instancias de si mesmo ao mesmo tempo;
- o *script* deve registrar um nome de máquina para um endereço de rede, mesmo que o DHCP não o tenha fornecido, usando para isso um padrão;
- o *script* deve registrar suas ações;
- o *script* deve registrar o endereço nos domínios direto e reverso;
- o registro reverso deve ser considerado em zonas com subredes de 24 bits;
- o *script* deve permitir execução pelo usuário não privilegiado do serviço DHCP.

A partir destes requisitos, foram desenvolvidos os algoritmos nas figuras seguir, tendo o *script* sido desenvolvido dividido em funções, facilitando a organização e reaproveitamento do código.

A linguagem de programação utilizada, *python* 3, permite dois modos básicos de programação, estruturada ou orientada a objeto. É usado um misto destas duas técnicas neste *script*, ora com orientação com objeto, quando da conexão ao *LDAP* e fornecimento deste objeto de conexão a várias funções, a montagem do objeto de inserção de dados no *LDAP* ou nas extrações de dados quando das consultas, ora com estruturação simples, no restante do *script*. Esta solução atende a todos os requisitos, permitindo o desenvolvimento e leitura quase

linear do código.

A opção por utilização de funções deve-se à possibilidade de redução de código e simplificação, pois algumas funções são usadas igualmente em várias partes da estrutura.

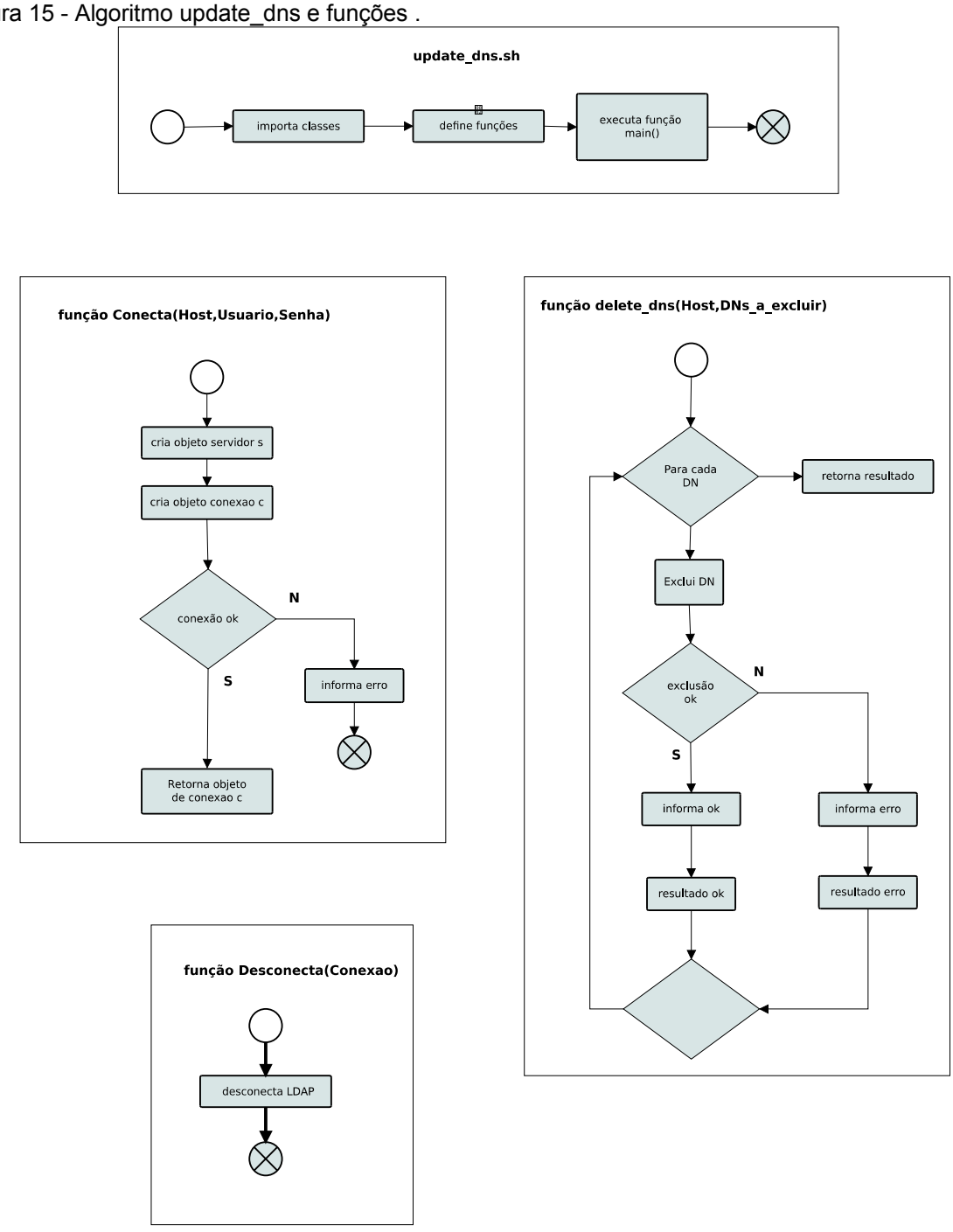
Conforme consulta às RFC's 1035, 2128 e 4034, foi decidido que os registros referentes às zonas reversas corresponderiam a subredes de 24 bits, com 254 *HOSTS*, gerando zonas com os três octetos da subrede, do tipo 0.168.192.IN-ADDR.ARPA, ou conforme os octetos das redes referidas na inserção. Desta forma, reduz-se o número possível de zonas em relação ao cadastro de uma zona reversa cheia para cada *HOST*, otimizando o espaço utilizado por dados pelo *OpenLDAP*.

Porém, observou-se um número maior de requisições ao *OpenLDAP* pelo ISC-BIND a cada consulta ao DNS, a taxa de 3/1, ou seja três consultas *LDAP* para cada consulta DNS. Com isso, deduz-se que em caso de serviço com previsão de alto tráfego, infere-se a necessidade de uso de servidores escravos intermediários, para *cache* dos dados, que reduziriam drasticamente o número de consultas ao *OpenLDAP*, armazenando os endereços de autoridade pelo período cadastrado para cada um destes na própria base de dados *LDAP*.

Além de aumentar exponencialmente a capacidade total do sistema resolvedor de nomes autoritativo e o isolamento do serviço recursivo, esta solução daria ao conjunto maior segurança na forma que as consultas seriam realizadas a servidores não diretamente ligados a base principal, conferindo uma camada a mais de isolamento entre os clientes potenciais fontes de ataques de negação de serviço distribuído (DDOS).

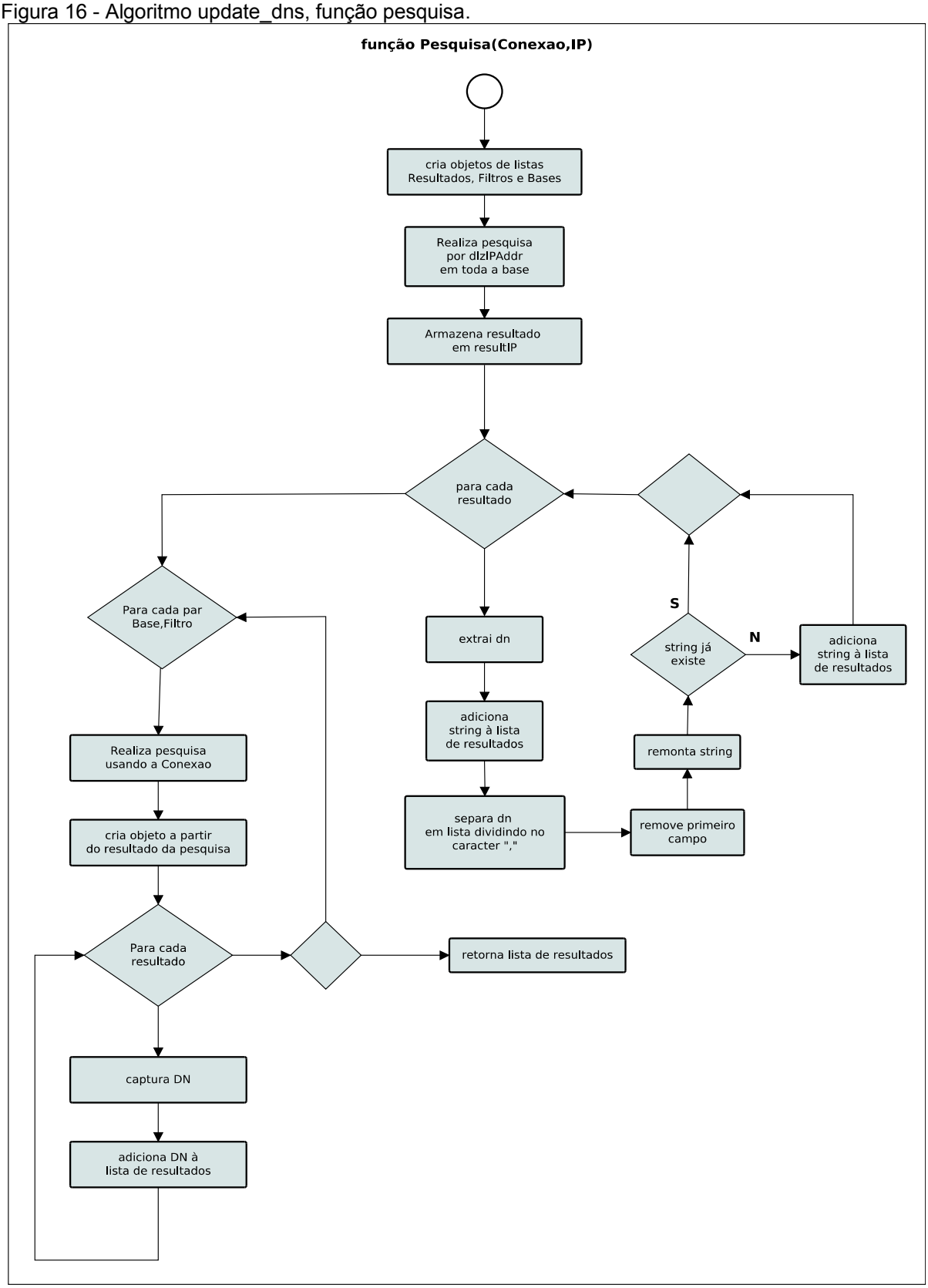
A queda de um servidor de *cache* intermediário pouco afetaria o conjunto, e ainda poderia ser contornada por monitoramento ativo do serviço com alguma inteligência de reinício ou isolamento de falhas, associada a detecção e tratamento de ataques deste tipo, sistemas de detecção e prevenção a intrusão.

Figura 15 - Algoritmo update_dns e funções .



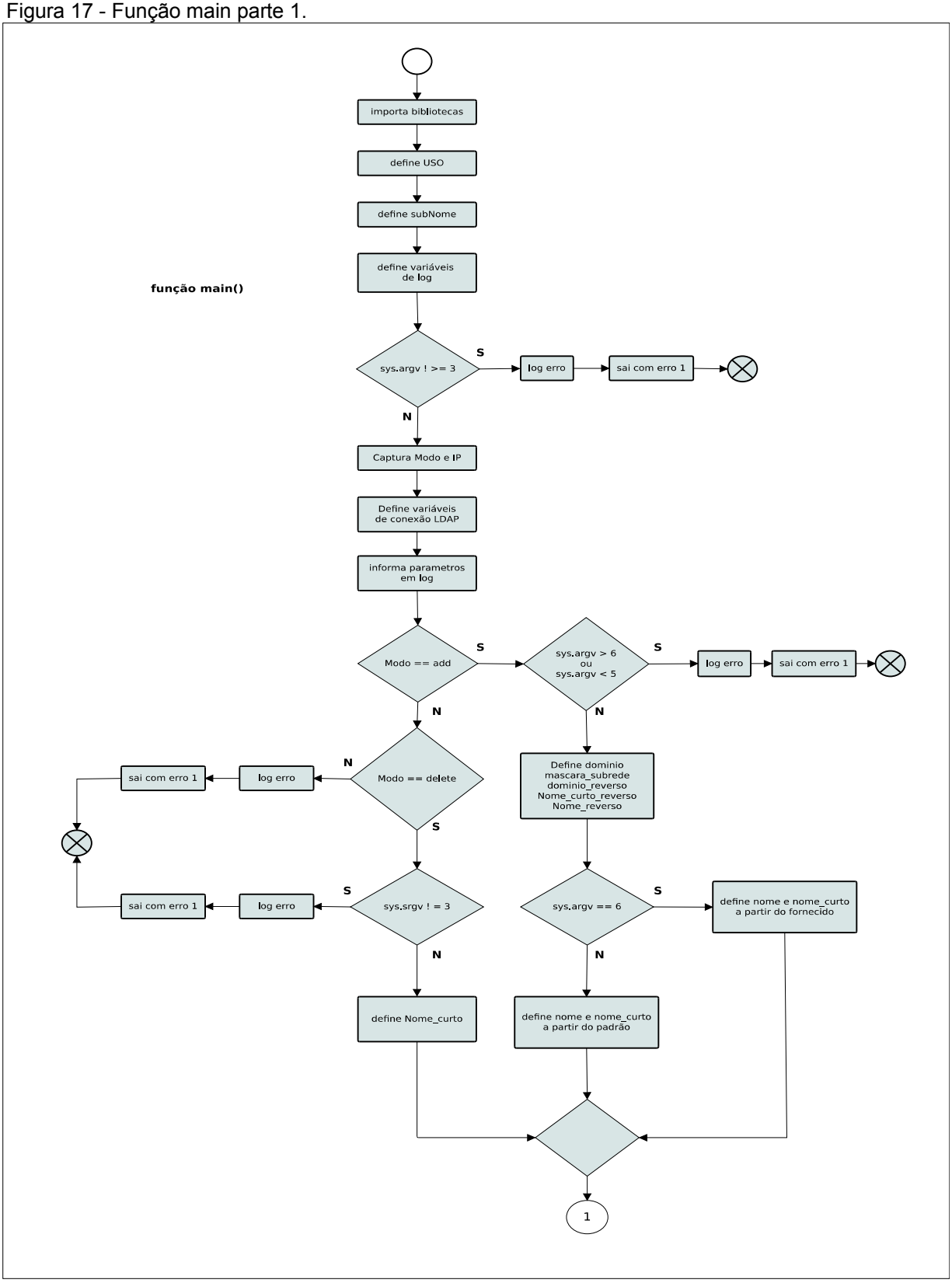
Fonte: autor.

Figura 16 - Algoritmo update_dns, função pesquisa.



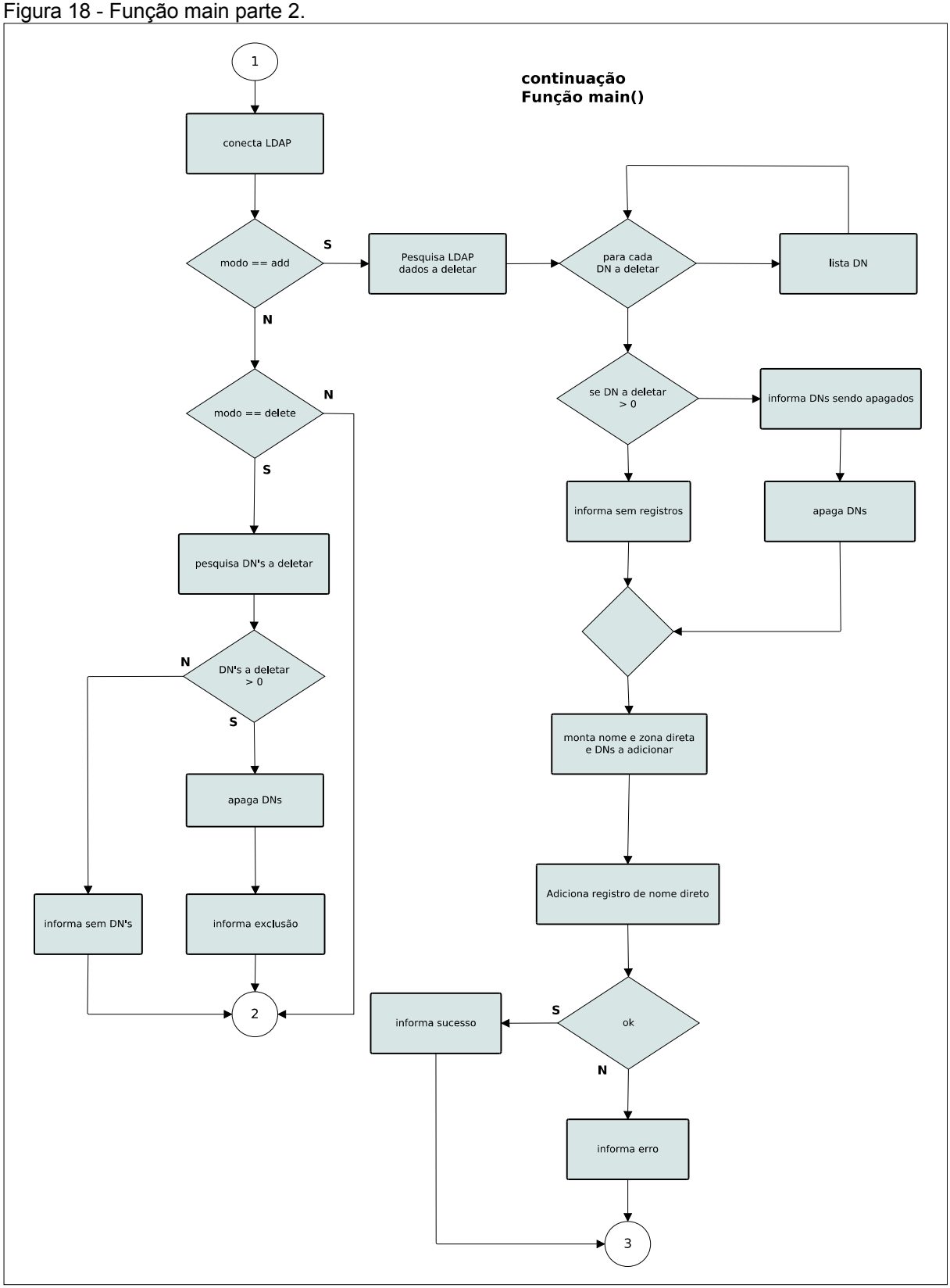
Fonte: Autor.

Figura 17 - Função main parte 1.



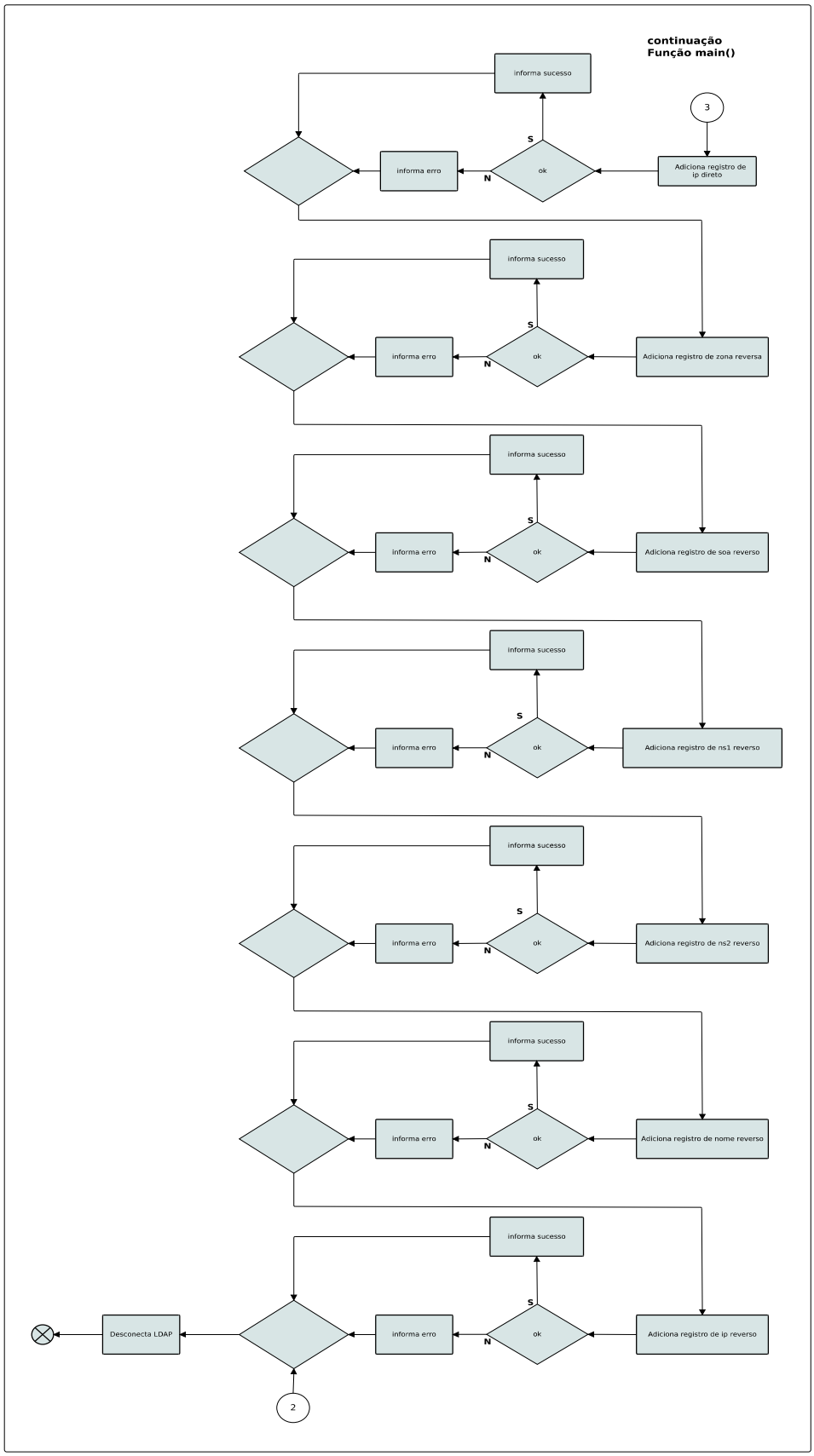
Fonte: Autor.

Figura 18 - Função main parte 2.



Fonte: Autor.

Figura 19 - Função main parte 3.



Fonte: Autor.

O *script* poderia ser construído em diversas linguagens de programação para atender aos requisitos acima. Foi escolhido pelo autor desenvolver em *Python* 3, já nativo na distribuição *ArchLinux*, adicionando a biblioteca *ldap*3, instalada pela ferramenta *pip* do pacote *python-pip*. Os comandos a seguir executam estas tarefas:

```
# pacman -Sy python-pip
# pip install ldap3
```

Assim, utilizando-se da biblioteca *ldap*3, foi desenvolvido o *script* em *python* 3, a seguir, conforme os algoritmos das figuras 15 a 19.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#
# recebe parametros $0 add 192.168.0.151 kirdeika.org 255.255.255.0 serv_prov
#                  delete 192.168.0.1
#$0 modo ip dominio mascara_de_rede [nome]
# import class and constants
import sys,os,logging
from ldap3 import Server, Connection, LEVEL, BASE, SUBTREE, ALL
def main():
 # texto mostrado caso seja encontrado erro de parametros
 USO = """\n===== Uso: update_dns.sh Modo IP Dominio Mascara_subrede [Nome]
                        Modo                = obrigatório, <add> para adicionar registros, <delete> para apagar.
                        IP                  = obrigatório, endereço IPv4 do registro. Ex. <192.168.0.151>
                        Dominio       = obrigatório em add, dominio ao qual o ip será registrado. Ex. <kirdeika.org>
                        Mascara_subrede = obrigatório em add, mascara da subrede. Ex. <255.255.255.0>
                        Nome                = opcional em add, nome do registro sem o dominio Ex. <www>
             Exemplos
                        update_dns.sh add 192.168.0.151 kirdeika.org 255.255.255.0
             update_dns.sh delete 192.168.0.151
             update_dns.sh add 192.168.0.1 kirdeika.org 255.255.255.0 domain
                        \n\n"""
 # padrao de nomes a serem adicionados no DNS
 SubNome = "dhcp-"
 # Nível de log a ser registrado - INFO , WARNING , ERROR, DEBUG
 # define o log
 logging.basicConfig(filename='/var/log/update_dns.log', level=logging.DEBUG, format='%(asctime)s %(levelname)s:
             %(message)s',  datefmt='%x %X ')
 if not len(sys.argv) >= 3:
  logging.error('===== Não foram recebidos todos os parametros, abandonando operação. Parametros %s tamanho
%s' % ( str(sys.argv), str(len(sys.argv)) ))
  sys.stderr.write('===== Parametros insuficientes, foram informados: ' + str(sys.argv) + "\n\n")
  sys.stderr.write(USO)
  sys.exit(1)
 else:
 # captura de parametros
  Modo = str(sys.argv[1]).strip()
  IP = str(sys.argv[2]).strip()
 # padrões deste script
 BaseDN = 'ou=dns,dc=kirdeika,dc=org'
 Usuario = 'cn=bind,dc=kirdeika,dc=org'
 HOST = 'localhost'
 Senha = 'senha'
 # o que foi passado
 logging.info("Parametros passados: %s. " % (str(sys.argv)))
```

```
# verificando parametros
if Modo == 'add' :
   if len(sys.argv) > 6 or len(sys.argv) < 5:
      logging.error("    Modo de adição selecionado, mas parametros insuficientes ou incorretos.")
      sys.stderr.write("    Modo de adição selecionado, mas parametros insuficientes ou incorretos.")
      sys.stderr.write(USO)
      sys.exit(1)
   else:
      Dominio = str(sys.argv[3]).strip()
      Mascara_subrede = str(sys.argv[4]).strip()
      Dominio_reverso = IP.split(".")[2] + "." + IP.split(".")[1] + "." + IP.split(".")[0] + ".IN-ADDR.ARPA"
      Nome_curto_reverso = IP.split(".")[3]
      Nome_reverso = IP.split(".")[3] + "." + Dominio_reverso
      if len(sys.argv) == 6:
        Nome_curto = str(sys.argv[5])
        Nome = Nome_curto + "." + Dominio
      else:
       Nome_curto = SubNome + IP.translate(str.maketrans(".","-"))
       Nome = SubNome + IP.translate(str.maketrans(".","-")) + "." + Dominio
      logging.debug("    Selecionado modo de adição de registros ->> Nome %s IP %s Dominio %s Dominio_reverso
%s." % (Nome,IP,Dominio,Dominio_reverso))
elif Modo == 'delete' :
   if not len(sys.argv) == 3 :
      logging.error("    Modo de exclusão selecionado, mas parametros insuficientes ou incorretos.")
      sys.stderr.write("    Modo de exclusão selecionado, mas parametros insuficientes ou incorretos.")
      sys.stderr.write(USO)
      sys.exit(1)
      logging.debug("    Selecionado modo de exclusão de registros ->> Nome %s IP %s" % (Subnome,IP))
   else:
      Nome_curto = SubNome + IP.translate(str.maketrans(".","-"))
else:
   logging.error("    Não foi selecionado um modo válido -> %s." % Modo)
   sys.stderr.write("    Não foi selecionado um modo válido -> %s." % Modo)
   sys.stderr.write(USO)
   exit(1)
# conectando no OpenLDAP
Conexao_LDAP = Conecta(HOST,Usuario,Senha)
# ação!
if Modo == 'add':
   DNs_a_deletar = pesquisa(Conexao_LDAP,IP)
   # mostrando as pesquisas
   for DN in DNs_a_deletar:
      logging.debug("    DN por nome: " + DN + "")
   if len(DNs_a_deletar) > 0 :
      logging.info("    Deletando " + str(len(DNs_a_deletar)) + " registros existentes anteriores")
      deleta_dns(Conexao_LDAP,DNs_a_deletar)
   else:
      logging.info("    Sem registros anteriores para deletar.")
   logging.debug("    Adicionando registros...")
   # adicionando endereços direto e reverso
   # montando registro direto a adicionar
   Nome_curto = str(Nome).split('.', maxsplit=1)[0]
   Zona_direta = str(Nome).split('.', maxsplit=1)[1]
   DN_novo_direto = "dlzHOSTName=" + Nome_curto + ",dlzZoneName=" + Zona_direta +
",ou=dns,dc=kirdeika,dc=org"
   DN_novo_direto_1 = "dlzRecordID=1,dlzHOSTName=" + Nome_curto + ",dlzZoneName=" + Zona_direta +
",ou=dns,dc=kirdeika,dc=org"
   # montando registro reverso a adicionar
   DN_novo_zona_reversa = "dlzZoneName=" + Dominio_reverso + ",ou=dns,dc=kirdeika,dc=org"
   DN_novo_reverso = "dlzHOSTName=" + Nome_curto_reverso + ",dlzZoneName=" + Dominio_reverso +
",ou=dns,dc=kirdeika,dc=org"
   DN_novo_reverso_soa = "dlzHOSTName=@,dlzZoneName=" + Dominio_reverso + ",ou=dns,dc=kirdeika,dc=org"
   DN_novo_reverso_soa_1 = "dlzRecordID=1,dlzHOSTName=@,dlzZoneName=" + Dominio_reverso +
",ou=dns,dc=kirdeika,dc=org"
   DN_novo_reverso_soa_2 = "dlzRecordID=2,dlzHOSTName=@,dlzZoneName=" + Dominio_reverso +
",ou=dns,dc=kirdeika,dc=org"
   DN_novo_reverso_soa_3 = "dlzRecordID=3,dlzHOSTName=@,dlzZoneName=" + Dominio_reverso +
",ou=dns,dc=kirdeika,dc=org"
   DN_novo_reverso_1 = "dlzRecordID=1,dlzHOSTName=" + Nome_curto_reverso + ",dlzZoneName=" +
Dominio_reverso + ",ou=dns,dc=kirdeika,dc=org"
   if Conexao_LDAP.add(DN_novo_direto, ['dlzHOST'], {'dlzHOSTName': Nome_curto}):
      logging.info("    Registro direto de HOST adicionado com sucesso")
```

```
    else:
        logging.info("    Erro na inserção de registro direto de HOST.")
        logging.debug(Conexao_LDAP.result)
    if Conexao_LDAP.add(DN_novo_direto_1, ['dlzARecord','top'], {'dlzHOSTName': Nome_curto, 'dlzIPAddr': IP,
'dlzRecordID': 1, 'dlzTTL': 3600, 'dlzType': 'a'}):
        logging.info("    Registro direto de ip adicionado com sucesso.")
    else:
        logging.info("    Erro na inserção de registro direto de ip.")
        logging.debug(Conexao_LDAP.result)
    # adicionando zona reversa
    if Conexao_LDAP.add(DN_novo_zona_reversa, ['dlzZone','top'], {'dlzZoneName': Dominio_reverso}):
        logging.info("    Registro de zona reversa adicionada com sucesso.")
    else:
        logging.info("    Erro na inserção de zona reversa, possivelmente já existe!")
        logging.debug(Conexao_LDAP.result)
    # adicionando registro SOA da zona reversa
    if Conexao_LDAP.add(DN_novo_reverso_soa, ['dlzHOST'], {'dlzHOSTName': '@'}):
        logging.info("    Registro reverso soa adicionado com sucesso.")
    else:
        logging.info("    Erro na inserção de registro soa reverso, possivelmente existe!")
        logging.debug(Conexao_LDAP.result)
    # adicionando registro SOA id 1 da zona reversa
    if Conexao_LDAP.add(DN_novo_reverso_soa_1, ['dlzSOARecord','top'], {'dlzAdminEmail':
'root.kirdeika.org.' ,'dlzExpire': 608400, 'dlzMinimum': 86400, 'dlzPrimaryNS': 'ns1.kirdeika.org.', 'dlzRefresh': 2800,
'dlzRetry': 7200, 'dlzSerial': 10, 'dlzHOSTName': '@', 'dlzRecordID': 1, 'dlzTTL': 86400, 'dlzType': 'soa'}):
        logging.info("    Registro reverso soa 1 (autoridade)  adicionado com sucesso.")
    else:
        logging.info("    Erro na inserção de registro soa reverso, possivelmente existe!")
        logging.debug(Conexao_LDAP.result)
    # adicionando registro SOA id 2 da zona reversa
    if Conexao_LDAP.add(DN_novo_reverso_soa_2, ['dlzNSRecord','top'], {'dlzData': 'ns1.kirdeika.org.', 'dlzHOSTName':
'@', 'dlzRecordID': 2, 'dlzTTL': 86400, 'dlzType': 'ns'}):
        logging.info("    Registro reverso soa ns1 adicionado com sucesso.")
    else:
        logging.info("    Erro na inserção de registro soa ns2 reverso, possivelmente existe!")
        logging.debug(Conexao_LDAP.result)
    # adicionando registro SOA id 3 da zona reversa
    if Conexao_LDAP.add(DN_novo_reverso_soa_3, ['dlzNSRecord','top'], {'dlzData': 'ns2.kirdeika.org.', 'dlzHOSTName':
'@', 'dlzRecordID': 3, 'dlzTTL': 86400, 'dlzType': 'ns'}):
        logging.info("    Registro reverso soa ns2 adicionado com sucesso.")
    else:
        logging.info("    Erro na inserção de registro soa ns2 reverso, possivelmente existe!")
        logging.debug(Conexao_LDAP.result)
    # adicionando nome reverso
    if Conexao_LDAP.add(DN_novo_reverso, ['dlzHOST'], {'dlzHOSTName': Nome_curto_reverso}):
        logging.info("    Registro reverso de HOST adicionado com sucesso.")
    else:
        logging.info("    Erro na inserção de registro reverso de HOST.")
        logging.debug(Conexao_LDAP.result)
    # adicionando endereço reverso
    if Conexao_LDAP.add(DN_novo_reverso_1, ['dlzPTRRecord'], {'dlzData': Nome + ".", 'dlzHOSTName':
Nome_curto_reverso, 'dlzRecordID': 1, 'dlzTTL': 3600, 'dlzType': 'ptr'}):
        logging.info("    Registro reverso de ip adicionado com sucesso.")
    else:
        logging.info("    Erro na inserção de registro reverso de ip.")
        logging.debug(Conexao_LDAP.result)
 if Modo == 'delete':
    DNs_a_deletar_delete = pesquisa(Conexao_LDAP,IP)
    if len(DNs_a_deletar_delete) > 0:
        logging.info("    Deletando " + str(len(DNs_a_deletar_delete)) + " registros existentes anteriores")
        deleta_dns(Conexao_LDAP,DNs_a_deletar_delete)
    else:
        logging.info("    Sem registros para deletar.")
 Desconecta(Conexao_LDAP)
 sys.exit(0)
 # fim do programa!!!
######Definição de funções em uso no programa
def Conecta(HOST,Usuario,Senha):
 # define the server
 s = Server(HOST=HOST, port=389, use_ssl=False)  # define an unsecure LDAP server, requesting info on DSE and
schema
 # define the connection
```

```
 c = Connection(s, user=Usuario, password=Senha, auto_bind='NONE', version=3, authentication='SIMPLE',
client_strategy='SYNC', auto_referrals=True, check_names=True, read_only=False, lazy=False,
raise_exceptions=False)
 # perform the Bind operation
 if not c.bind():
    logging.error('    Servidor LDAP inacessível... ', c.result)
    exit(2)
 else:
    return c
def Desconecta(c):
 c.unbind()
# função de exclusão da base
def deleta_dns(Conexao,DNs_a_excluir):
 for DN in DNs_a_excluir:
    if Conexao.delete(DN):
        result = True
        logging.debug("    Exclusão de " + DN + " realizada com sucesso.")
    else:
        result = False
        logging.warning("    Houve erro na exclusão de " + DN + " Verifique.")
        logging.debug(Conexao.result)
 return result
# função de pesquisa de registros, retorna lista de dn's dos achados por nome curto e IP
def pesquisa(Conexao, IP):
 # define variaveis e listas
 lista_retorno = []
 Base = "ou=dns,dc=kirdeika,dc=org"
 Filter = "(dlzIPAddr=" + IP + ")"
 Filters = []
 Bases = []
 IP_separado = IP.split(".")
 A = IP_separado[0].strip("[\']")
 B = IP_separado[1].strip("[\']")
 C = IP_separado[2].strip("[\']")
 D = IP_separado[3].strip("[\']")
 Filters.append("(dlzHOSTName=" + D + ")")
 Bases.append("dlzZoneName=" + C + "." + B + "." + A + ".IN-ADDR.ARPA,ou=dns,dc=kirdeika,dc=org")
 Filters.append("(dlzHOSTName=" + D + "." + C + ")")
 Bases.append("dlzZoneName=" + B + "." + A + ".IN-ADDR.ARPA,ou=dns,dc=kirdeika,dc=org")
 Filters.append("(dlzHOSTName=" + D + "." + C + "." + B + ")")
 Bases.append("dlzZoneName=" + A + ".IN-ADDR.ARPA,ou=dns,dc=kirdeika,dc=org")
 # pesquisa no LDAP por IP
 Conexao.search(search_base = Base,
        search_filter = Filter,
        search_scope = SUBTREE,
        attributes = ['dn','dlzHOSTName'])
 result_IP = Conexao.response
 # processa resultados por IP
 for entry in result_IP:
    DN = str(entry['dn'])
    lista_retorno.append(DN)
    DN_pai = DN.split(",")
    del DN_pai[0]
    DN = ','.join(DN_pai)
    if DN not in lista_retorno:
        lista_retorno.append(DN)
 # agora vou pesquisar nomes reversos, podem ser para a.b.c.d
 # dlzHOSTName=d,dlzZoneName=c.b.a.IN-ADDR.ARPA ou
 # dlzHOSTName=d.c,dlzZoneName=b.a.IN-ADDR.ARPA ou
 # DlzHOSTName=d.c.b,dlzZoneName=a.IN-ADDR.ARPA
 # pesquisa por nomes
 for Filter,Base in zip(Filters,Bases):
  Conexao.search(search_base = str(Base).strip("[\']"),
         search_filter = str(Filter).strip("[\']"),
         search_scope = SUBTREE,
         attributes = ["dn"],)
  result = Conexao.response
  # processa resultados por nomes, recursivamente
  for entry in result:
     DN = str(entry['dn'])
     lista_retorno.append(DN)
 return sorted(lista_retorno, reverse=True)
```

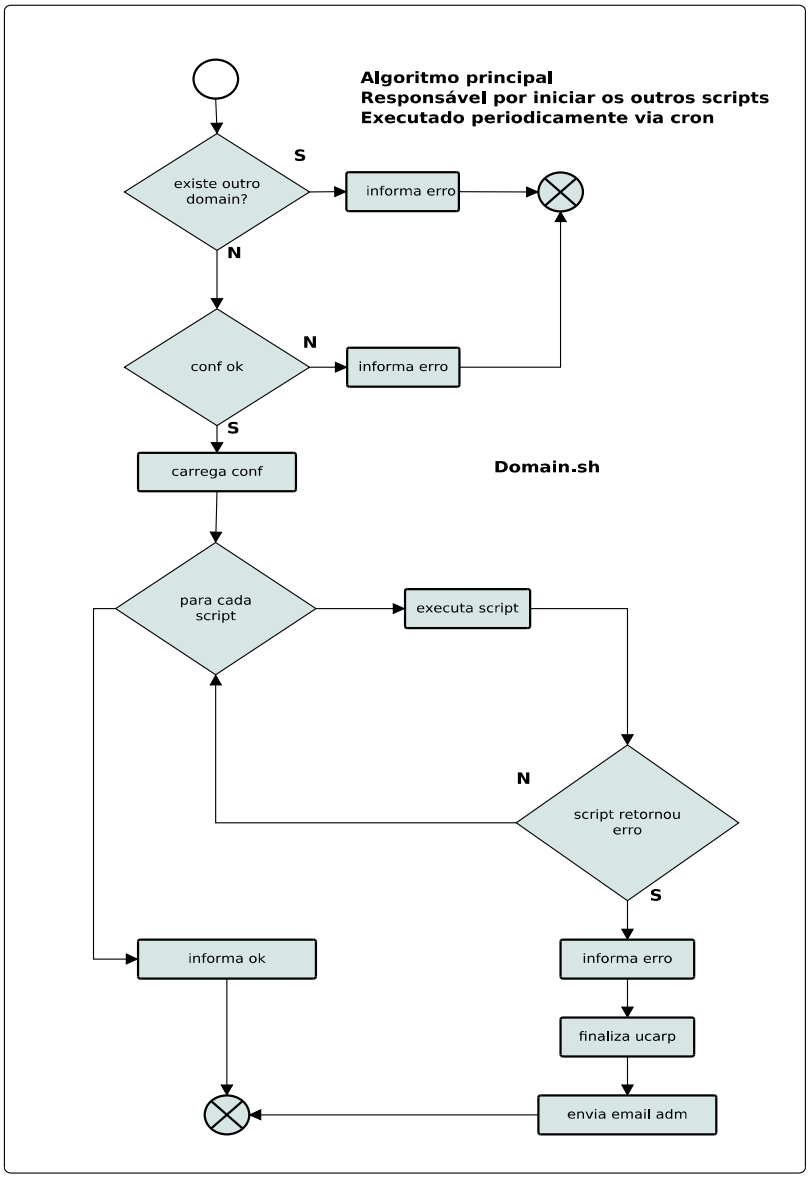
```
# programa principal
if __name__ == "__main__":
 main()
```

Pode-se observar as ações do *script* pelo arquivo de *log* em /var/log/update_dns.log, e o nível de *log* inicialmente é definido em DEBUG, devendo-se alterar para INFO em produção, após verificada toda a funcionalidade. Observa-se também que, pela estrutura e decisão de projeto, não serão deletadas as zonas reversas criadas, ou as diretas existentes. As zonas diretas devem existir, e o seu cadastro antecipado, pela ferramenta de administração do *OpenLDAP* instalada ou diretamente via arquivos LDIF. Em caso da zona reversa existir, um erro não crítico informará a situação, podendo seguramente ser ignorado.

2 Domain

Este é o principal *script* da solução, sendo ao mesmo tempo o mais simples, derivado diretamente das políticas e do algoritmo da solução de alta disponibilidade. Para seu desenvolvimento foi usado *Shell script*, cujo código resultante tem agendada sua execução seguindo as políticas do capítulo 11, utilizando-se do recurso de agendamento nativo na distribuição *ArchLinux* cron. Todas as configurações da solução ficam armazenadas em arquivo único, em /etc/domain.conf, com as devidas permissões de acesso a apenas o usuário root. O algoritmo da solução está representado na figura 20.

Figura 20 - Algoritmo Domain.



Fonte: Autor.

Note-se que todos os *scripts* são executados como usuário root. O código resultante referente ao algoritmo da figura 20 está a seguir.

```
#!/bin/bash
#
#
#Copyright (c) 2015, Mario Sergio KirdeiKa Junior
#All rights reserved.
#
#Redistribution and use in source and binary forms, with or without modification, are
permitted provided that the following conditions are met:
#
#1. Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.
#
#2. Redistributions in binary form must reproduce the above copyright notice, this list of
```

```
conditions and the following disclaimer in the documentation and/or other materials provided
with the distribution.
#
#3. Neither the name of the copyright holder nor the names of its contributors may be used to
endorse or promote products derived from this software without specific prior written
permission.
#
#THIS software IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS software, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
#
#
# este script chama cada script de teste da solução, e finaliza UCARP em caso de falha em
algum serviço
# dessa forma, o(s) outro(s) servidor(es) disponíveis na solução assumem os serviços
#
#
# variáveis do CONF
# SERVIDOR            = nome de máquina da conexão
# SENHA               = senha da conexão
# LDAP_login          = login LDAP
# BASE_LDAP           = domínio base do LDAP
PIDOF=$(whereis -b pidof | awk '{print $2}')
SYSTEMCTL=$(whereis -b systemctl | awk '{print $2}')
UCARP=$(whereis -b UCARP | awk '{print $2}')
KILL=$(whereis -b kill | awk '{print $2}')
LOG=/var/log/domain.log
CONF=/etc/domain.conf
DIR_scripts="/root/ha"
scripts="recupera_LDAP.sh recupera_DNS.sh recupera_DHCP.sh recupera_UCARP.sh"
resultado=0
# verifica a configuração
if [ -f $CONF ] ; then
        . $CONF
else
        echo -ne "\n`date`\nO arquivo de configuração $CONF é inválido!\n"  >> $LOG
       exit 5
fi
if [ $($PIDOF $0 | wc -w) -ne 0 ] ; then
       echo "`date` : Existe outra instância de $0 em execução! Saindo..." >> $LOG
       exit 1
fi
for i in $scripts ; do
       $DIR_scripts/$i
       result=$?
       if [ $result -ne 0 ] ; then
              echo "`date` : O script $i retornou $result. Finalizando o UCARP..." >> $LOG
              $KILL -9 $($PIDOF UCARP) 2>&1 >> $LOG
              break
       fi
done
echo "`date` : Teste da solução de HA, resultado final $result." >> $LOG
exit 0
```

E a seguir está o arquivo /etc/domain.conf, que contém todas as configurações necessárias aos *scripts* da solução.

```
# configuração dos scripts da solução de HA
# configuração dos scripts da solução de HA
# servidor serv_a.kirdeika.org
EMAILADM=kirdeikajr@gmail.com
USER_EMAIL=kirdeikajr@gmail.com
SERVER_EMAIL=smtp.google.com
PASS_EMAIL="senha"
SERVIDOR=serv_a.kirdeika.org
ID_CLUSTER=1
IP_CLUSTER=192.168.0.1
SENHA="senha"
IP_SERVIDOR=192.168.0.10
INTERFACE=eth0
PESO=1
SYSLOG=local5
script_UP=/usr/bin/vip-up.sh
script_DOWN=/usr/bin/vip-down.sh
UCARP_LOG=/var/log/UCARP.log
LDAP_login="cn=domain,dc=kirdeika,dc=org"
REC_LDAP_LOG=/var/log/recupera_LDAP.log
REC_UCARP_LOG=/var/log/recupera_UCARP.log
REC_DNS_LOG=/var/log/recupera_DNS.log
REC_DHCP_LOG=/var/log/recupera_DHCP.log
BASE_LDAP="dc=kirdeika,dc=org"
```

Para execução periódica deste *script*, deve ser instalado o pacote cronie, que provê o agendador de tarefas cron. Os comandos para instalação, habilitação e inicialização do serviço estão a seguir.

```
# pacman -Sy cronie
# systemctl enable cronie
# systemctl start cronie
```

A tabela de execução do cron para o usuário root (*crontab*), deve ficar como a seguir, aplicando a execução do *script* uma vez por minuto, usando o comando “crontab -e”:

```
* * * * * /root/ha/domain.sh
```

Conforme a figura 20, primeiro é verificado se há outro *script* em execução, em caso negativo é verificada a validade do arquivo de configuração. Se *ok*, cada *script* de verificação independente é executado. Caso algum encontre problemas, o que é verificado através do código de saída do *script*, padronizado em 0 para saída normal ou diferente de zero para problemas, o *script* interrompe a execução normal e finaliza o *UCARP* para retirar o servidor do *cluster*.

Esta atitude é tomada pois cada *script* individualmente tenta recuperar uma vez o serviço para o qual foi criado, e a decisão em caso de falha de algum

script é a retirada do mesmo do *cluster*, isolando o mesmo e avisando o administrador, se possível. É altamente aconselhável que seja implementado um sistema de verificação externo ao sistema, como um dos mencionados no capítulo 11.

3 Recupera_*UCARP*

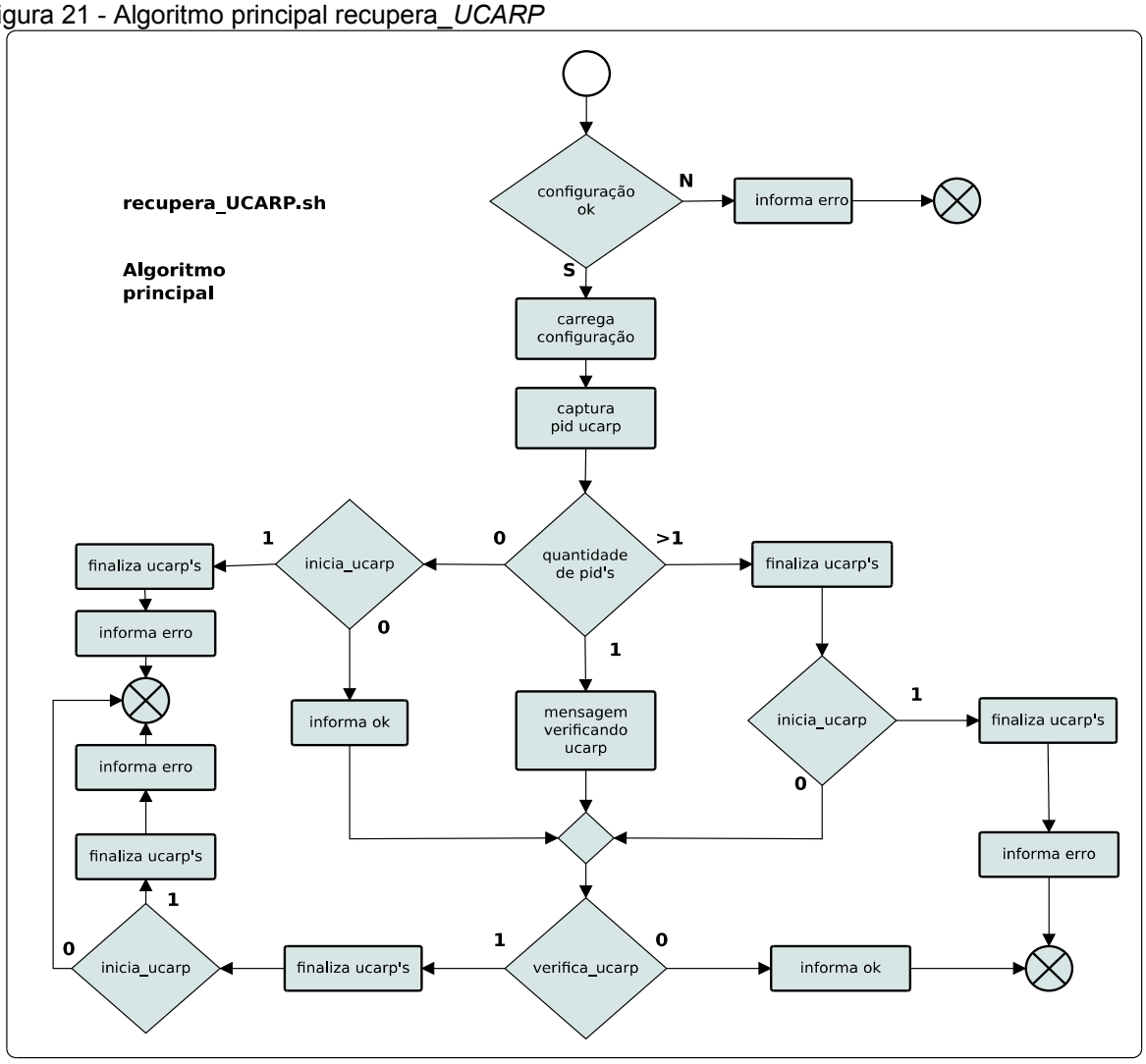
Este *script* é responsável por verificar, e se necessário tentar recuperar uma vez, a execução do *UCARP* e portanto manter ou remover o servidor do nó em caso de falha não recuperável do *UCARP*.

O *script* baseia-se na funcionalidade do *UCARP* de informar em seu arquivo de *log* o seu estado através de sinais enviados ao processo. O sinal SIGUSR1 enviado ao processo do *UCARP* faz com que o seu estado, *Master* ou *backup*, seja informado no *log*. Esta informação é procurada através da informação constituída de “ano/mes/dia hora:minuto:segundo” do momento em que foi enviado o sinal ao processo.

Se encontrado, é assumido que o *UCARP* entendeu o sinal, está ativo e pode se comunicar. Portanto, pode estabelecer eleição no *cluster*, assumindo *backup* ou *Master* conforme a necessidade. Em caso de falha, se houver *id* de processo de *UCARP*, é finalizado o mesmo, ou quantos *id's* forem, e um *UCARP* é iniciado com os parametros necessários. Uma nova verificação é realizada, e se *ok*, o *script* finaliza com erro 0, ou outro código de saída em caso de qualquer falha.

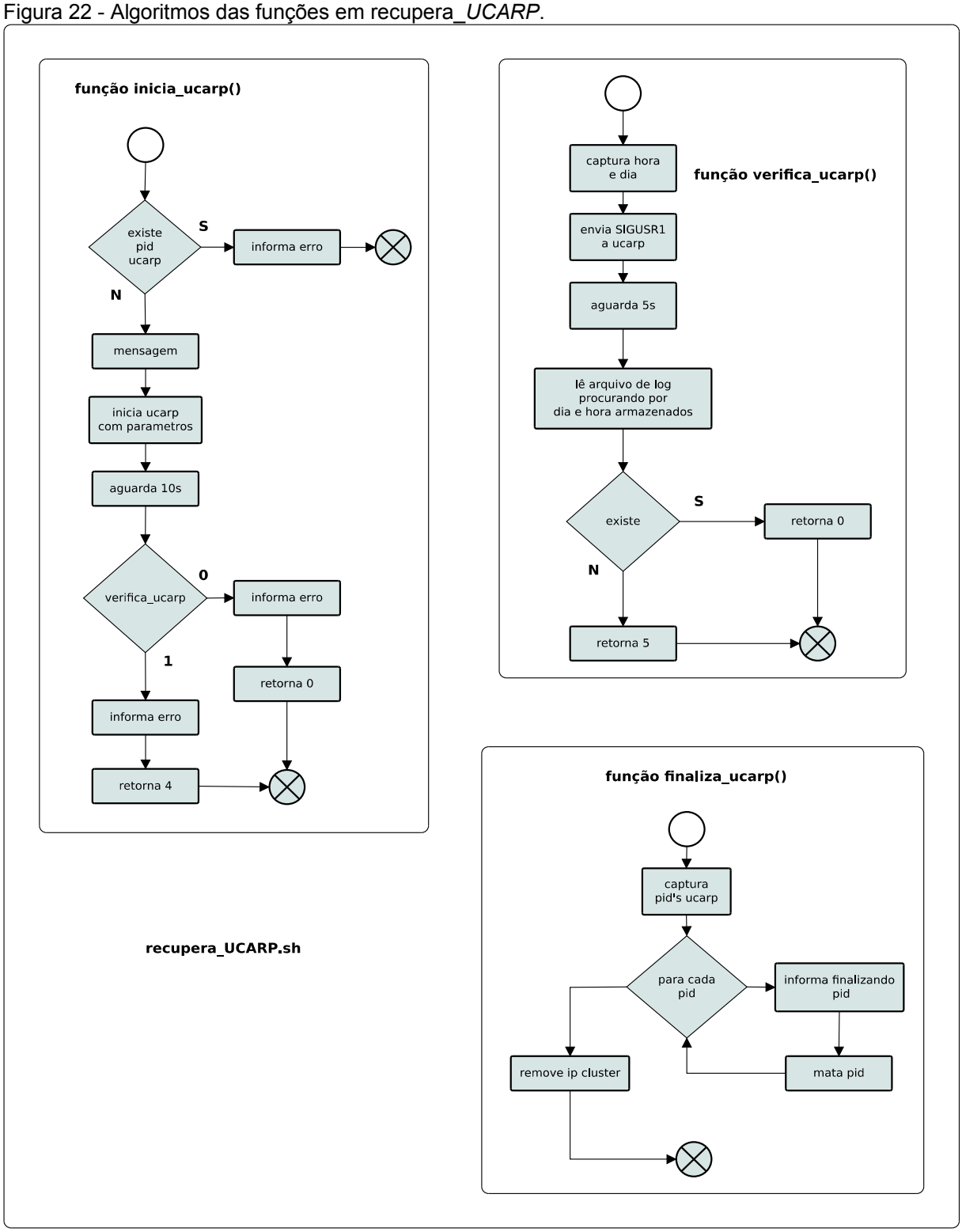
O algoritmo criado está nas figuras 21 e 22, e foi desenvolvido usando *Shell script*. Foi usado o arquivo de configuração da solução para armazenar os parâmetros necessários.

Figura 21 - Algoritmo principal recupera_*UCARP*



Fonte: Autor.

Figura 22 - Algoritmos das funções em recupera_*UCARP*.



Fonte: Autor.

O código resultante dos algoritmos das figuras 21 e 22 está a seguir.

```
#!/bin/bash

# recupera :UCARP: -> projeto de alta disponibilidade de recursos em redes -> mario sergio
#
# este script verifica a situação do UCARP, se necessário inicia ou reinicia o UCARP conforme parametros do projeto
#
# pós-graduação uniceub
#
# capta as variáveis de configuração
#
# SERVIDOR      = nome do servidor
# ID_CLUSTER    = endereço do cluster HA
# SENHA         = senha comum com os servidores do cluster
# IP_SERVIDOR   = endereço de rede deste servidor
# IP_CLUSTER    = endereço de rede a ser manipulado no HA
# INTERFACE     = interface de rede dos endereços envolvidos
# PESO          = valor de relevância deste servidor no cluster - quanto menor maior a prioridade no HA
# SYSLOG        = facilidade do syslog de captura dos log's do UCARP
# script_UP     = script a ser executado quando o UCARP promove o servidor a Master no cluster
# script_DOWN   = script a ser executado quando o UCARP depromove o servidor de Master no cluster, ( passa a
backup)
# UCARP_LOG     = arquivo do log do UCARP, vem da configuração do syslog-ng
# REC_UCARP_LOG = arquivo de log do script
#
PIDOF=$(whereis -b pidof | awk '{print $2}')
UCARP=$(whereis -b UCARP | awk '{print $2}')
KILL=$(whereis -b kill | awk '{print $2}')
CONF=/etc/domain.conf
# scripts
finaliza_UCARPs() {
        PIDS_UCARP=$($PIDOF UCARP)
        for i in $PIDS_UCARP ; do
                echo "`date`: Finalizando UCARP pid $i" >> $REC_UCARP_LOG
                $KILL -9 $i 2>&1 >> $REC_UCARP_LOG
                /usr/bin/ip addr delete $IP_CLUSTER/16 dev $INTERFACE
        done
}
inicia_UCARP() {
        if [ $($PIDOF UCARP | wc -w) -gt 0 ] ; then
                echo "`date` : Existem instâncias de UCARP executando\! Não posso iniciar outro."  >>
$REC_UCARP_LOG
                echo 1
        else
                echo "`date` : Iniciando UCARP." >> $REC_UCARP_LOG
                $UCARP -i $INTERFACE -s $IP_SERVIDOR -v $ID_CLUSTER -p $SENHA -a $IP_CLUSTER -b 1 -k
$PESO -u $script_UP -d $script_DOWN -r 2 -B -f $SYSLOG 2>&1 >>$REC_UCARP_LOG
                sleep 3
                if [ $( verifica_UCARP ) -eq 0 ] ; then
                        echo "`date` : UCARP inicializado ok!" >> $REC_UCARP_LOG
                        echo 0
                else
                        echo "`date` : Não foi possível iniciar o UCARP." >> $REC_UCARP_LOG
                        echo 4
                fi
        fi
}
verifica_UCARP() {
        AGORA=$(date "+%Y/%m/%d %H:%M:%S")
        kill -s SIGUSR1 $($PIDOF UCARP) 2>&1 >/dev/null
        sleep 1
#       echo "pesquisa $AGORA " >> $REC_UCARP_LOG
#       grep  "$AGORA" $UCARP_LOG >> $REC_UCARP_LOG
        grep -q "$AGORA" $UCARP_LOG 2>&1 >/dev/null
        if [ $? -eq 0 ] ; then
                echo 0
        else
                echo 5
        fi
}
if [ -f $CONF ] ; then
        . $CONF
else
```

```
            echo -ne "\n`date`\nO arquivo de configuração $CONF é inválido!\n"  >> $REC_UCARP_LOG
            exit 1
fi
PID_UCARP=$($PIDOF UCARP)
if [ $(echo $PID_UCARP | wc -w) -eq 0 2>&1 >/dev/null ] ; then
            if [ $( inicia_UCARP ) -ne 0 ] ; then
                        echo "`date` : Não foi possível iniciar o UCARP....."  >> $REC_UCARP_LOG
                        exit 6
            fi
elif [ $(echo $PID_UCARP | wc -w) -gt 1 2>&1 >/dev/null ] ; then
            finaliza_UCARPs
            if [ $( inicia_UCARP ) -ne 0 ] ; then
                        echo "`date` : Não foi possível iniciar o UCARP....."  >> $REC_UCARP_LOG
                        exit 7
            fi
else
            echo "`date` : Verificando o UCARP....."  >> $REC_UCARP_LOG
fi
if [ $( verifica_UCARP ) -eq 0 ] ; then
            echo "`date` : UCARP ok!"  >> $REC_UCARP_LOG
            exit 0
else
            echo "`date` : O UCARP não responde, tentando recuperar....."  >> $REC_UCARP_LOG
            finaliza_UCARPs
            if [ $( inicia_UCARP ) -ne 0 ] ; then
                        echo "`date` : Não foi possível iniciar o UCARP....."  >> $REC_UCARP_LOG
                        exit 8
            fi
fi
exit 0
```

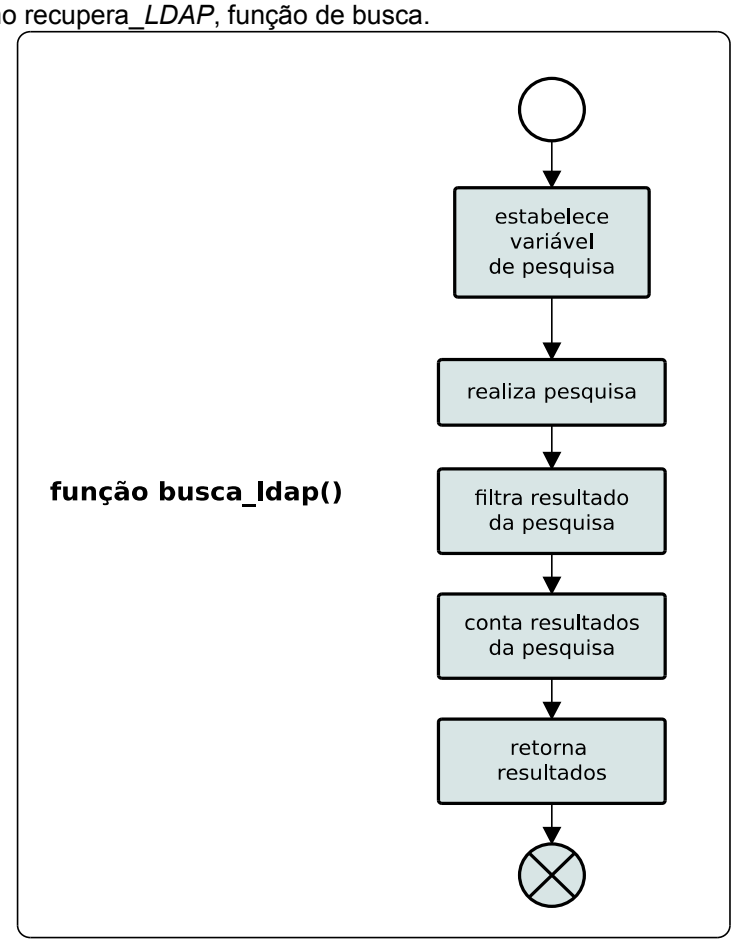
4 Recupera_*LDAP*

Este *script* é responsável por testar, e se necessário tentar reiniciar o servidor *OpenLDAP* uma vez. O teste é realizado tentando uma busca pelo parâmetro cn do usuário de *login* criado especificamente para esta operação, usando credenciais armazenadas no arquivo de configuração da solução.

Em caso de falha na busca, um erro é informado e o serviço *LDAP* é reiniciado uma vez. Se o reinício foi realizado com sucesso, uma nova busca é realizada, e se novamente houver falha, o *script* finaliza com erro diferente de 0. No caso da primeira ou após o reinício a busca resultar positiva, o *script* finaliza com erro 0.

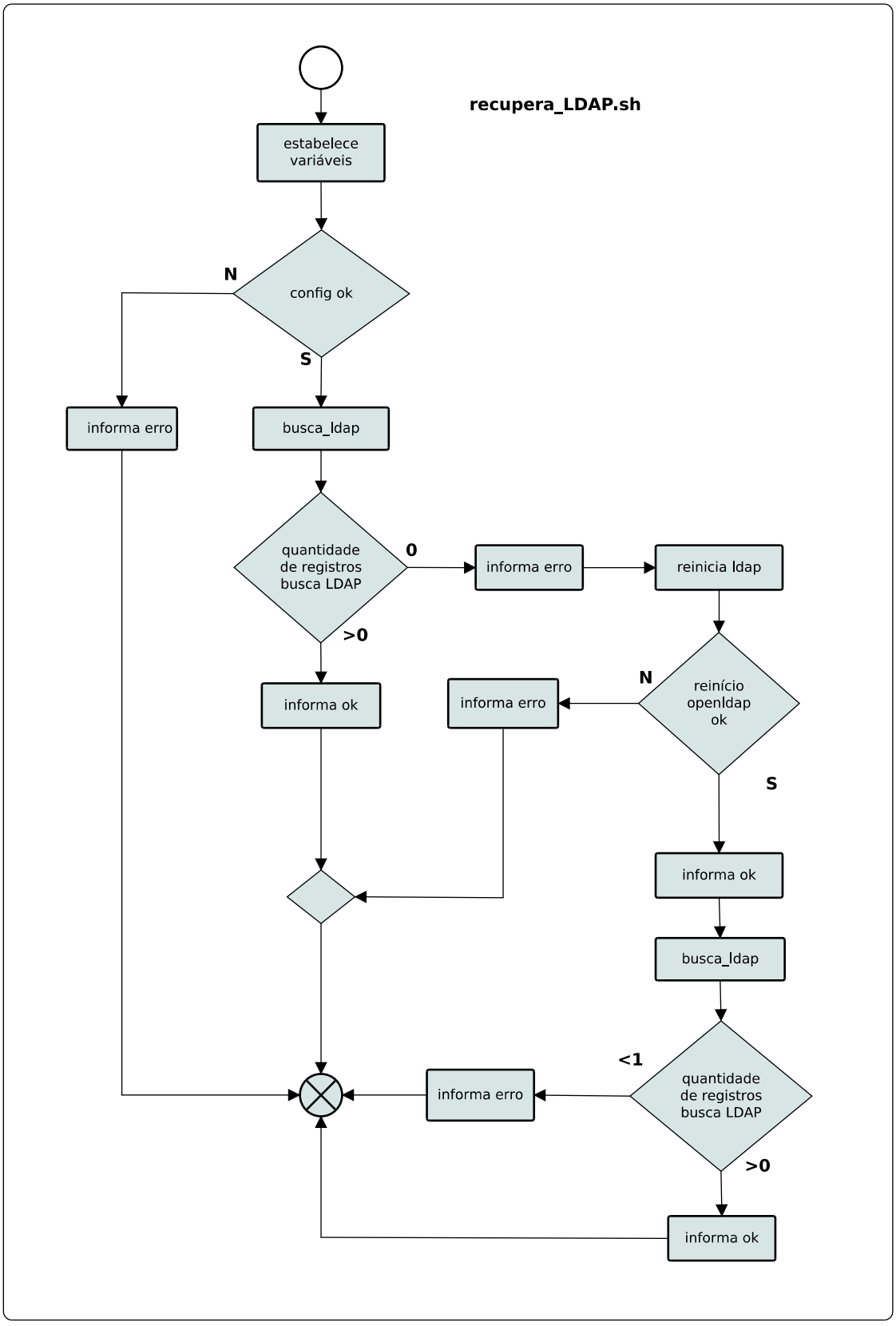
O algoritmo resultante está nas figuras 23 e 24 a seguir.

Figura 23 - Algoritmo recupera_*LDAP*, função de busca.



Fonte: Autor.

Figura 24 - Algoritmo principal recupera_*LDAP*.



Fonte: Autor.

O código resultante em *Shell script* dos algoritmos expostos nas figuras 23 e 24 está a seguir:

```
#!/bin/bash
#
#
#Copyright (c) 2015, Mario Sergio KirdeiKa Junior
#All rights reserved.
#
#Redistribution and use in source and binary forms, with or without modification, are
permitted provided that the following conditions are met:
#
#1. Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.
#
#2. Redistributions in binary form must reproduce the above copyright notice, this list of
conditions and the following disclaimer in the documentation and/or other materials provided
with the distribution.
#
#3. Neither the name of the copyright holder nor the names of its contributors may be used to
endorse or promote products derived from this software without specific prior written
permission.
#
#THIS software IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS software, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
#
#
# este script testará o slapd através de conexão ssl e bind, via ldapsearch, retornando o cn
do usuário de login
#
# usa login e senha do arquivo domain.conf
#
# variáveis do CONF
# SERVIDOR            = nome de máquina da conexão
# SENHA               = senha da conexão
# LDAP_login          = login LDAP
# BASE_LDAP           = domínio base do LDAP
PIDOF=$(whereis -b pidof | awk '{print $2}')
SYSTEMCTL=$(whereis -b systemctl | awk '{print $2}')
ldapsearch=$(whereis -b ldapsearch | awk '{print $2}')
KILL=$(whereis -b kill | awk '{print $2}')
LOG=/var/log/recupera_LDAP.log
CONF=/etc/domain.conf
# verifica a configuração
if [ -f $CONF ] ; then
        . $CONF
else
        echo -ne "\n`date`\nO arquivo de configuração $CONF é inválido!\n"  >> $LOG
        exit 5
fi
busca_LDAP() {
       procura_por="cn=$(echo $LDAP_login | cut -d, -f 1 | cut -d= -f 2)"
       result=$($ldapsearch -D $LDAP_login -w "${SENHA}" -H "ldaps://$SERVIDOR:636/" -b
$BASE_LDAP $procura_por cn 2>/dev/null | grep -v ^# | grep -v ^$ | grep $procura_por | wc -l)
       echo $result
}
if [ $(busca_LDAP) -eq 0 ] ; then
```

```
        echo "`date` : Encontrada falha no OpenLDAP! Reiniciando... " >> $REC_LDAP_LOG
        $SYSTEMCTL restart slapd >> $REC_LDAP_LOG
        if [ $? -ne 0 ] ; then
                echo -ne "\n`date` : Não foi possível reiniciar o OpenLDAP! Verifique a
seguir.\n\n" >> $REC_LDAP_LOG
                $SYSTEMCTL status slapd >> $REC_LDAP_LOG
                exit 3
        else
                echo "`date` : Reiniciado o OpenLDAP OK!" >> $REC_LDAP_LOG
                if [ $(busca_LDAP) -lt 1 ] ; then
                        echo "`date` : O OpenLDAP foi reiniciado, mas a busca falhou novamente!"
>> $REC_LDAP_LOG
                        exit 2
                else
                        echo "`date` : O OpenLDAP foi reiniciado, busca OK!" >> $REC_LDAP_LOG
                        exit 0
                fi
        fi
else
        echo "`date` : Busca OK!" >> $REC_LDAP_LOG
        exit 0
fi
```

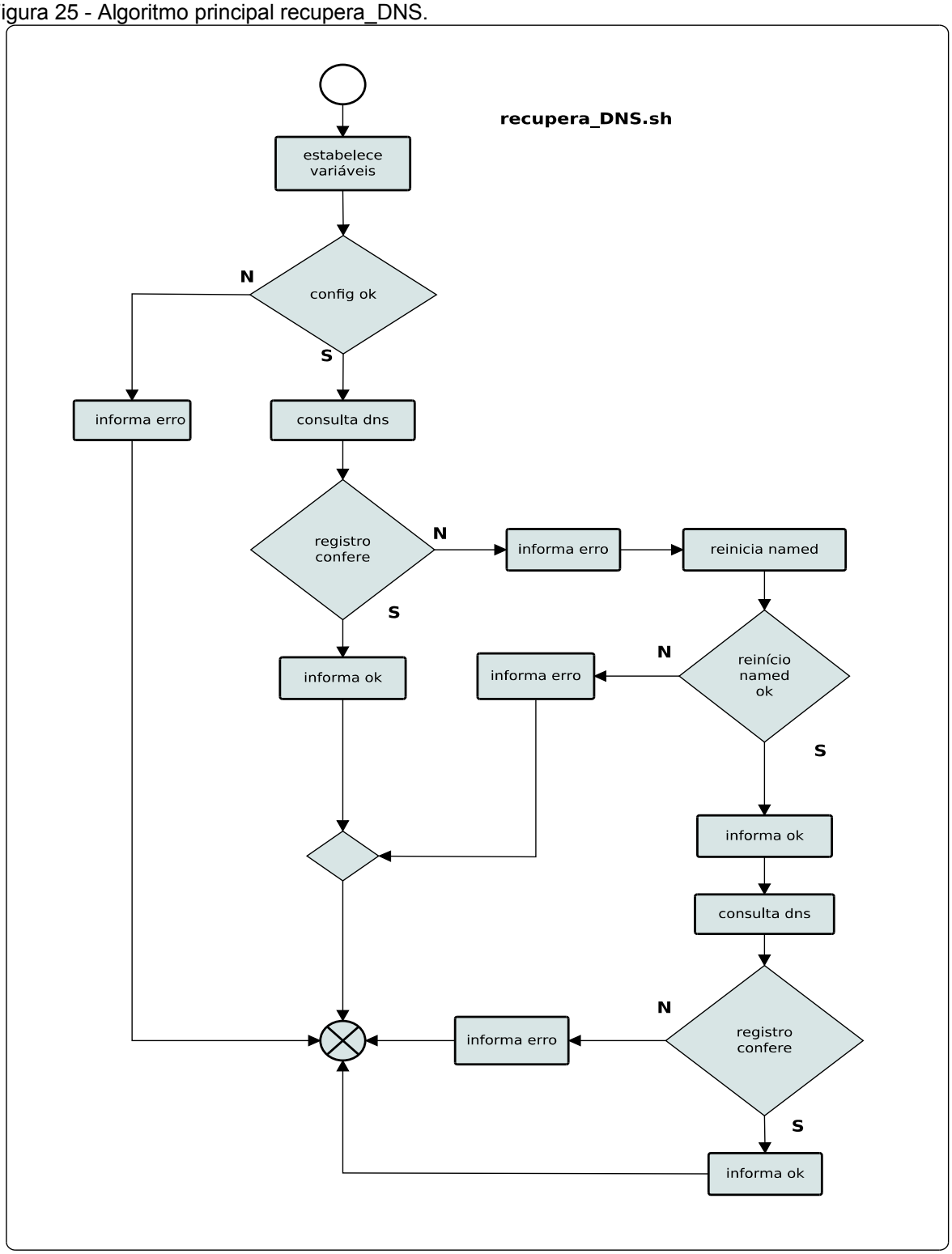
5 Recupera_DNS

Este *script* é responsável por testar e se necessário reiniciar uma vez o serviço de resolvedor de nomes. É usada a ferramenta *dig*, provida pelo pacote ISC-BIND, executando duas consultas a um registro de teste. É usado o arquivo de configuração global da solução, em /etc/domain.conf. Uma consulta é realizada ao servidor local de nomes, apontando para o endereço IP fixo e se o *UCARP* estiver em *Master*, para o IP do *cluster*.

Se o valor configurado na variável SERVIDOR conferir com o retornado na consulta ao serviço presente no endereço da variável IP_CHECK, a consulta é considerada bem sucedida. Em caso de falha, o serviço *named* é reiniciado uma vez, e caso falhe o reinício ou a consulta novamente, o *script* é finalizado com erro diferente de zero. Em caso da primeira da consulta após o reinício retornem ok, o *script* é finalizado com erro 0.

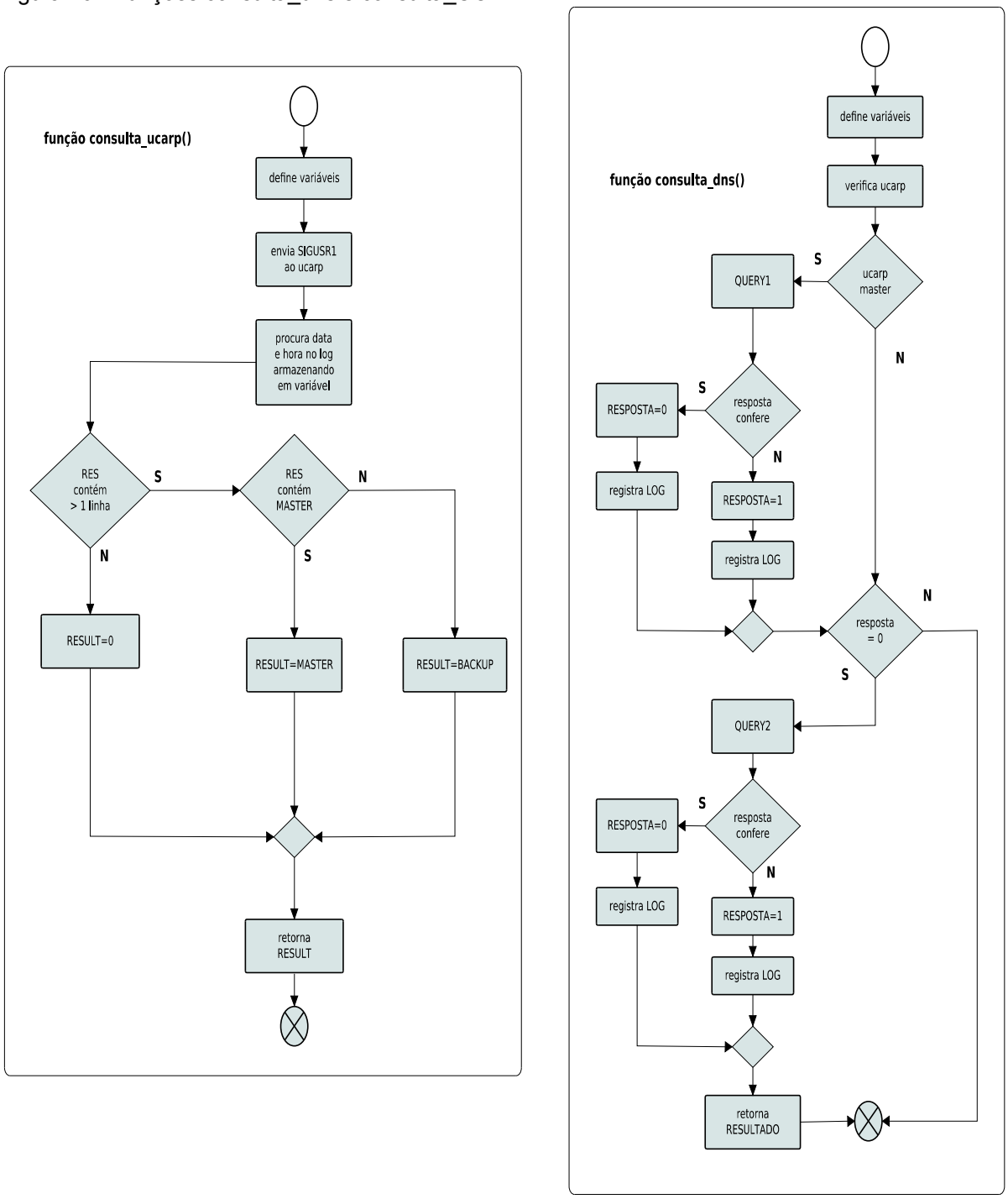
O algoritmo resultante está nas figuras 25 e 26 a seguir.

Figura 25 - Algoritmo principal recupera_DNS.



Fonte: Autor.

Figura 26 - Funções consulta_dns e consulta_*UCARP*.



Fonte: Autor.

O código resultante dos algoritmos descritos nas figuras 25 e 26 está a seguir:

```
#!/bin/bash
#Copyright (c) 2015, Mario Sergio KirdeiKa Junior
#All rights reserved.
#
#Redistribution and use in source and binary forms, with or without modification, are
permitted provided that the following conditions are met:
#
#1. Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.
#
#2. Redistributions in binary form must reproduce the above copyright notice, this list of
conditions and the following disclaimer in the documentation and/or other materials provided
with the distribution.
#
#3. Neither the name of the copyright holder nor the names of its contributors may be used to
endorse or promote products derived from this software without specific prior written
permission.
#
#THIS software IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS software, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
# este script testará o DNS através de consulta aos ips IP_SERVIDOR e IP_CLUSTER
# usa login e senha do arquivo domain.conf
PIDOF=$(whereis -b pidof | awk '{print $2}')
SYSTEMCTL=$(whereis -b systemctl | awk '{print $2}')
BIN_HOST=$(whereis -b HOST | awk '{print $2}')
KILL=$(whereis -b kill | awk '{print $2}')
CONF=/etc/domain.conf
IP_CHECK=192.168.0.10
QUERY1=""
QUERY2=""
# verifica a configuração
if [ -f $CONF ] ; then
        . $CONF
else
        echo -ne "\n`date`\nO arquivo de configuração $CONF é inválido!\n"  >> $REC_DNS_LOG
        exit 5
fi
consulta_UCARP() {
       if [ $(pidof UCARP | wc -w) -gt 0 ] ; then
                AGORA=$(date "+%Y/%m/%d %H:%M:%S")
               kill -s SIGUSR1 $($PIDOF UCARP)
                RES=$(grep "$AGORA" $UCARP_LOG)
               if [ $(echo $RES | wc -l) -gt 0 ] ; then
                       if [ $(echo $RES | grep Master | wc -l) -gt 0 ] ; then RESULT=Master ;
else RESULT=backup ; fi
                else
                        RESULT=0
                fi
       else
               RESULT=0
       fi
       echo $RESULT
}
consulta_dns() {
       RESPOSTA=0
       if [ $(consulta_UCARP) == "Master" ] ; then
               QUERY1=$(dig @$IP_CLUSTER $SERVIDOR +short +tries=1 +retries=1 +time=1
+norecurse 2>/dev/null | grep -v ^\; | grep -E "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}")
               if [ ${QUERY1} == ${IP_CHECK} 2>/dev/null ] ; then
                       RESPOSTA=0
                       MENSAGEM="Master"
               else
```

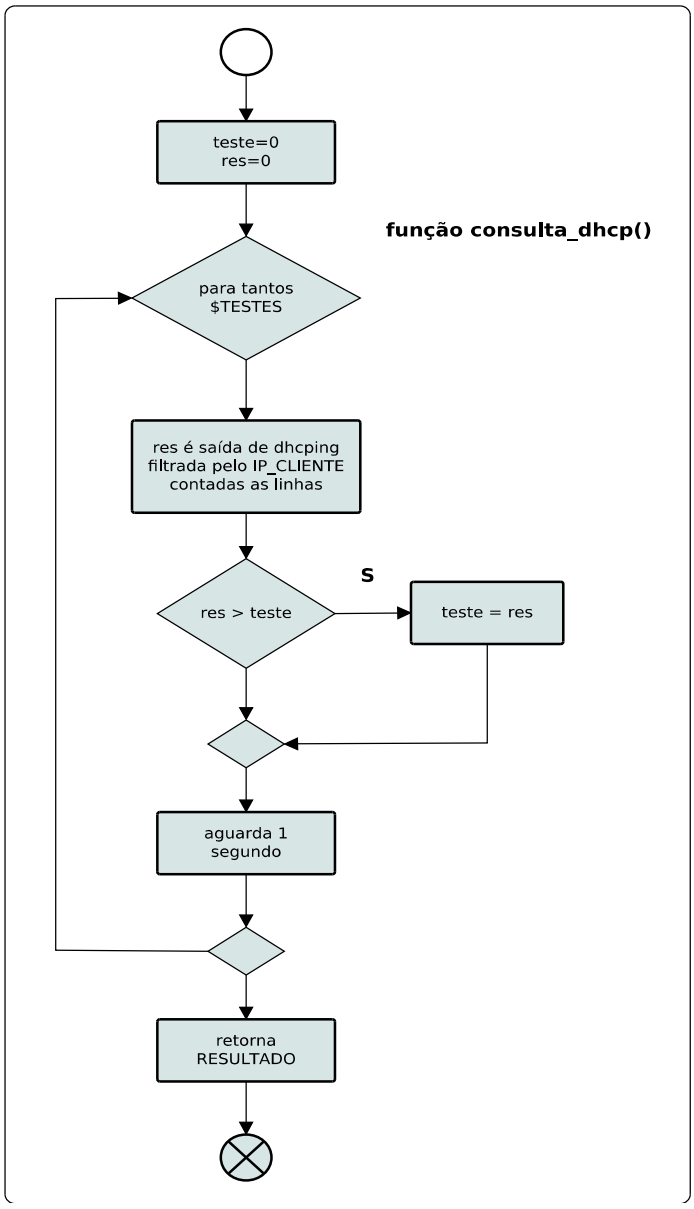
```
                        echo "`date` : Consulta $IP_CLUSTER sem sucesso..." >> $REC_DNS_LOG
                        RESPOSTA=1
                fi
        else
                MENSAGEM="não Master"
        fi
        if [ $RESPOSTA -eq 0 ] ; then
                QUERY2=$(dig @$IP_SERVIDOR $SERVIDOR +short +tries=1 +retries=1 +time=1
+norecurse 2>/dev/null | grep -v ^\; | grep -E "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]
{1,3}")
                if [ ${QUERY2} == $IP_CHECK 2>/dev/null ] ; then
                        RESPOSTA=0
                else
                        echo "`date` : Consulta $IP_SERVIDOR sem sucesso..." >> $REC_DNS_LOG
                        RESPOSTA=1
                fi
        fi
        echo "${RESPOSTA} ${MENSAGEM}"
}
read RESPOSTA MENSAGEM < <(consulta_dns)
if [ $RESPOSTA -ne 0  ] ; then
        systemctl restart named >> $REC_DNS_LOG
        if [ $? -ne 0 ] ; then
                echo "`date` : Não foi possível reiniciar o named!" >> $REC_DNS_LOG
                exit 2
        else
                echo "`date` : Reiniciado o named!" >> $REC_DNS_LOG
                read RESPOSTA MENSAGEM < <(consulta_dns)
                if [ $RESPOSTA -ne 0 ] ; then
                        echo "`date` : Foi reiniciado o named porém sem sucesso, não é possível
consultar nomes!" >> $REC_DNS_LOG
                        exit 2
                else
                        echo "`date` : Foi reiniciado o named, é possível consultar nomes!" >>
$REC_DNS_LOG
                fi
        fi
else
        echo "`date` : Verificação DNS ok. Estado = $MENSAGEM ." >> $REC_DNS_LOG
fi
exit 0
```

6 Recupera_DHCP

Este *script* é responsável por realizar uma solicitação de endereço aos servidores DHCP, aguardando por resposta do serviço instalado localmente. Caso não seja recebida uma resposta do servidor local, o serviço é reiniciado uma vez e nova solicitação é realizada. Em caso de resposta que coincida com um endereço previamente cadastrado especialmente para o teste, o *script* é finalizado sem erro. Em caso de falha na resposta por não coincidir com o endereço previamente cadastrado ou não haver resposta, mesmo após a reinicialização do serviço, o *script* é finalizado com erro.

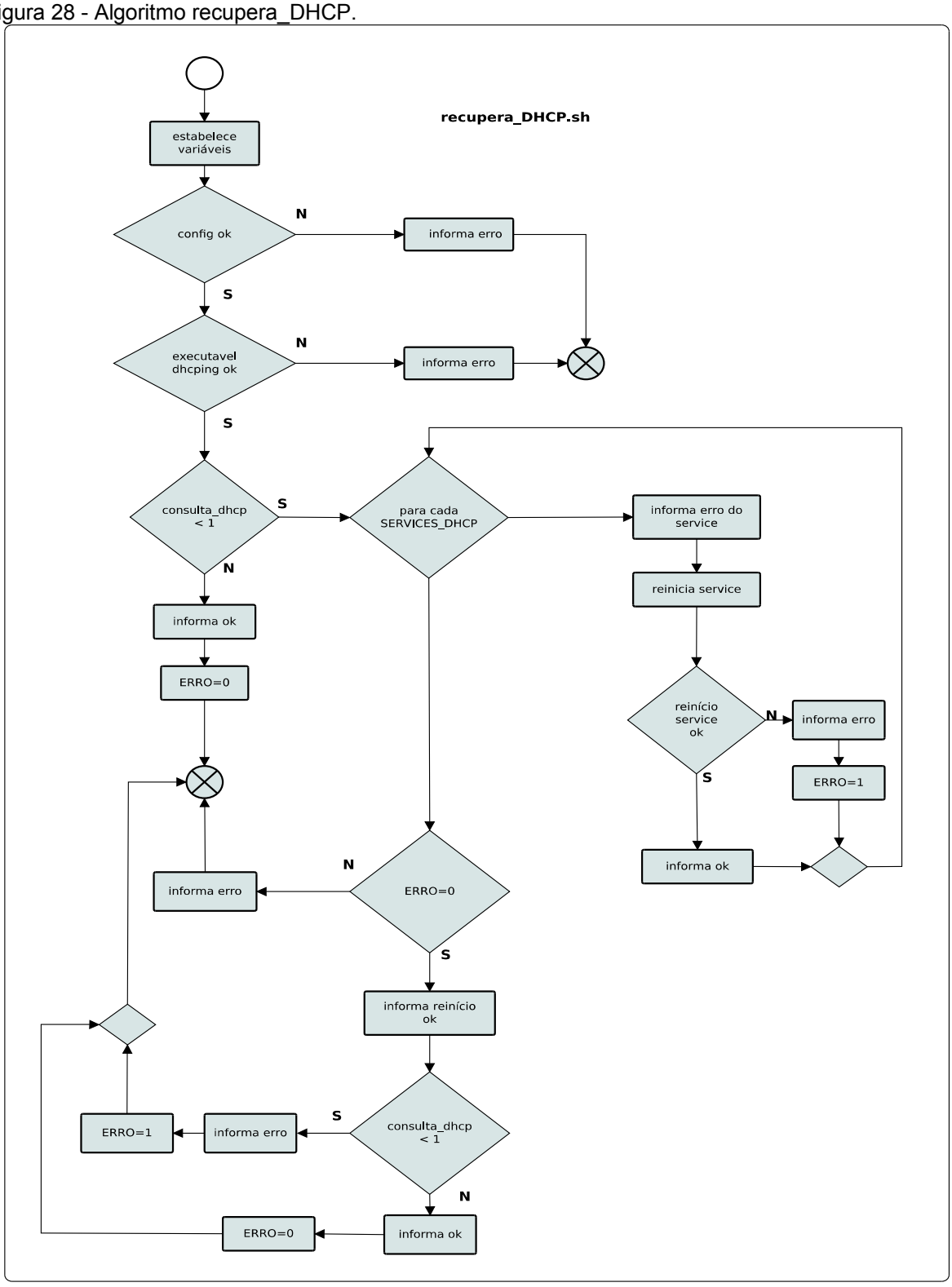
As figuras 27 e 28 a seguir descrevem os algoritmos utilizados.

Figura 27 - Algoritmo recupera_DHCP função consulta_dhcp.



Fonte: Autor

Figura 28 - Algoritmo recupera_DHCP.



Fonte: Autor

O código resultante dos algoritmos descritos pelas figuras 27 e 28 está a seguir:

```
#!/bin/bash
#
#
#Copyright (c) 2015, Mario Sergio KirdeiKa Junior
#All rights reserved.
#
#Redistribution and use in source and binary forms, with or without modification, are
permitted provided that the following conditions are met:
#
#1. Redistributions of source code must retain the above copyright notice, this list of
conditions and the following disclaimer.
#
#2. Redistributions in binary form must reproduce the above copyright notice, this list of
conditions and the following disclaimer in the documentation and/or other materials provided
with the distribution.
#
#3. Neither the name of the copyright holder nor the names of its contributors may be used to
endorse or promote products derived from this software without specific prior written
permission.
#
#THIS software IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS
OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE
GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS software, EVEN IF ADVISED
OF THE POSSIBILITY OF SUCH DAMAGE.
#
#
# este script testará o DHCP realizando uma solicitação dhcp de teste na interface eth2
#
# usa login e senha do arquivo domain.conf
#
PIDOF=$(whereis -b pidof | awk '{print $2}')
SYSTEMCTL=$(whereis -b systemctl | awk '{print $2}')
dhcping=$(whereis -b dhcping | awk '{print $2}')
KILL=$(whereis -b kill | awk '{print $2}')
CONF=/etc/domain.conf
DEV="eth0"
TESTES=3
SERVICES_DHCP="dhcpd4@eth0 dhcpd4@eth1"
MAC=$(ip link list dev $DEV | awk '/link/ {print $2}')
IP_CLIENTE=192.168.0.10
IP_DHCP=192.168.0.20
ERRO=0
# verifica a configuração
if [ -f $CONF ] ; then
        . $CONF
else
        echo -ne "\n`date`\nO arquivo de configuração $CONF é inválido!\n"  >> $REC_DHCP_LOG
        exit 5
fi
# verifica executável base dhcping
if [ ! -x $dhcping ] ; then
       echo -ne "\n`date`\nO executável dhcping não está disponível ou é inválido!\n"  >>
$REC_DHCP_LOG
        exit 5
fi
consulta_dhcp() {
       teste=0
       res=0
       for i in $(seq $TESTES) ; do
               res=$($dhcping -i -t 1 -c $IP_CLIENTE -s 255.255.255.255 -h $MAC 2>/dev/null |
grep $IP_CLIENTE | wc -l)
               if [ $res -gt $teste ] ; then teste=$res ; fi
               sleep 1
       done
       echo $teste
```

```
}
if [ $(consulta_dhcp) -lt 1 ] ; then
       for s in $SERVICES_DHCP ; do
       echo "`date` :  Teste do serviço de DHCP falhou, tentando reiniciar $s ..." >>
$REC_DHCP_LOG
               $SYSTEMCTL restart $s 2>&1 >/dev/null
               if [ $? -gt 0 ] ; then
                       ERRO=1
                       echo "`date` :  Reinício do serviço DHCP $s falhou. " >> $REC_DHCP_LOG
               else
                       echo "`date` :  Reinício do serviço DHCP $s ok, será realizado novo
teste. " >> $REC_DHCP_LOG
               fi
       done
       if [ $ERRO -eq 0 ] ; then
               echo "`date` :  Reinício do serviço DHCP $s ok, realizando novo teste. " >>
$REC_DHCP_LOG
               if [ $(consulta_dhcp) -lt 1 ] ; then
                       echo "`date` :  Reinício do serviço DHCP $s sem sucesso. " >>
$REC_DHCP_LOG
                       ERRO=1
               else
                       echo "`date` :  Reinício do serviço DHCP $s com sucesso. " >>
$REC_DHCP_LOG
                       ERRO=0
               fi
       fi
else
       echo "`date` :  Teste ok do serviço de DHCP." >> $REC_DHCP_LOG
       ERRO=0
fi
exit $ERRO
```

Pode-se observar pelo código, que o *script* baseia-se na funcionalidade do programa *dhcping*, o qual realiza uma solicitação de endereço a um servidor DHCP que pode ser informado como parâmetro, porém sem concluir a configuração de qualquer interface com o endereço fornecido.

Para a instalação do *dhcping*, usa-se a ferramenta *packer-color*, instalando o *dhcping* presente no *Arch User Repository* (AUR). O comando "*packer-color* -S *dhcping*", executado como usuário comum instala a ferramenta. Não é aconselhável usar o *packer* e derivados como usuário root.

Foi percebido em testes que se a solicitação partisse do mesmo servidor informado como alvo, a consulta não era bem sucedida. Assim, decidiu-se fazer a solicitação a um servidor diverso do local, aguardando até o final do *timeout* informado como parâmetro ao *dhcping*, e com esta condição o servidor local também respondia à solicitação, podendo então as respostas serem comparadas com a informação previamente definida como base, analisando-se e verificando-se.

O serviço DHCP pode ser configurado para servir endereços separadamente em cada interface, em cada subinterface ou subredes em cada uma das anteriores. Foi previsto o fornecimento de endereços por interface, mas em caso de necessidade, uma outra possível abordagem seria a configuração de interfaces separadas para cada rede, podendo usar protocolo 802.1Q para cada, na interface do servidor ou conectando a interface sem o referido protocolo a redes com este protocolo virtualmente configurado no *HOST*, assim, redes virtuais ou *VLAN's* podem ter servidores DHCP separados. Cada servidor que tenha que ser reiniciado está configurado na variável SERVICES_DHCP, mas não são testados um a um, pois foi constatado em prática que se um serviço DHCP cai, nenhum endereço é fornecido, havendo indisponibilidade em todos os serviços DHCP. Assim, uma interface ou serviço é suficiente para o teste, tendo como alvo um dos servidores do *cluster*, diverso em relação ao local.

Com estas condições acima atendidas, o endereço de teste do serv_a de escolha é o do serv_b. Caso o serv_a não responda após a consulta (ocorra timeout), os serviços DHCP são todos reiniciados uma vez, e em caso de falha na reinicialização ou nova falha em consulta, o *script* finaliza com erro.

APÊNDICE D – AUTORIDADE CERTIFICADORA LOCAL

Para os efeitos deste trabalho, a contratação de uma autoridade certificadora externa para o fornecimento dos certificados necessários ao funcionamento dos servidores inviabilizaria o projeto. Ao passo que, a criação e a manutenção de uma Autoridade Certificadora (CA) local, tanto serve aos propósitos deste trabalho como é algo útil às necessidades de criptografia e identificação interna de qualquer empresa.

Serão mostrados os passos necessários à criação de uma CA local, como se cria chaves privadas, solicitações de assinatura de certificados e a própria assinatura de certificados, utilizando a ferramenta *OpenSSL*.

Recomenda-se que os passos necessários à criação da CA sejam executados em máquina diferente dos servidores mencionados no restante deste trabalho, por motivos óbvios de confidencialidade, integridade e disponibilidade, autenticidade e não-repúdio. Ainda, que seja realizado *backup* periódico e consistente de todos os arquivos gerados, ou da própria máquina virtual onde os passos serão realizados. Este *backup* deve ser criptografado, por questão de confidencialidade.

1 Criação Da Autoridade Certificadora

Em termos de Infraestrutura de chaves públicas, a Autoridade Certificadora (CA), consiste na entidade confiável por todos os entes envolvidos. Esta confiança se baseia em métodos conhecidos e auditados de verificação de identidade. Em termos práticos, é aquela entidade que detém uma chave privada, com a qual se assinam digitalmente as solicitações de assinatura de certificado (*csr*) geradas a partir das chaves privadas das entidades solicitantes, após a verificação de sua identidade. Estas *csr's* assinadas, os certificados digitais, são então usados para atestar a veracidade das identidades dos envolvidos.

Criar uma CA consiste tecnicamente em criar um par de chaves, uma privada e uma pública, assinar a pública com o uso da privada, manter a confidencialidade da privada e divulgar a quem interessar a pública. Para os propósitos deste trabalho, a chave pública será instalada conforme necessário em todos os servidores envolvidos, serão geradas as chaves privadas da CA e dos servidores, conforme necessário, assinados os certificados usando os requisitos necessários, e instalados os certificados conforme necessário.

Para a criação da CA, basta uma máquina com *Linux*, notadamente *ArchLinux* neste trabalho, com acesso às redes onde se localizam as máquinas clientes para efeitos de praticidade de cópia criptografada pela rede usando a ferramenta *scp*, e a instalação do pacote *OpenSSL*. Cria-se primeiro o diretório que conterá a CA, então instala-se o *OpenSSL*, com os comandos a seguir.

```
# mkdir -m 700 /root/POS_CEUB
# pacman -Sy OpenSSL
```

Então, com todo o trabalho seguinte no diretório criado, procede-se à criação da chave privada, do tipo RSA, com tamanho 8192bits, criptografada com AES256, com o comando a seguir, sendo solicitada senha:

```
# openssl genrsa -aes256 -out POS_CEUB.key 8192
```

Após, o certificado raiz da CA pode ser criado, com os devidos parâmetros fornecidos pelo processo interativo que será iniciado, usando a chave acima, conforme o comando a seguir:

```
# openssl req -x509 -new -key POS_CEUB.key -days 3650 -out POS_CEUB.pem
Enter pass phrase for POS_CEUB.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:BR
State or Province Name (full name) [Some-State]:DISTRITO FEDERAL
Locality Name (eg, city) []:BRASILIA
Organization Name (eg, company) [Internet Widgits Pty Ltd]:POS_CEUB
Organizational Unit Name (eg, section) []:Redes com enfase em Segurança
Common Name (e.g. server FQDN or YOUR name) []:CA POS_CEUB
Email Address []:kirdeikajr@gmail.com
```

Com isso, o certificado raiz da CA POS_CEUB.pem foi gerado. Este certificado será usado para assinar as *csr's* dos servidores e usuários, assim como será instalado em todas as máquinas para validação da cadeia de confiança.

2 Certificados Das Máquinas Clientes

Então, a partir de uma máquina onde seja necessário a criação de um certificado, pode-se gerar uma nova chave privada, uma *csr*, copiar-se esta csr para a máquina onde está a chave privada e certificado da CA, assinar este certificado e então copiar de volta e instalar o certificado assinado onde este seja necessário. Este seria o procedimento normal em caso de uma CA externa. No caso deste trabalho, as chaves, *csr's* e certificados podem ser criadas na máquina CA, então copiados os arquivos necessários às máquinas de destino.

O certificado do *OpenLDAP* necessário a este trabalho tem uma peculiaridade. Cada certificado valida um servidor através do campo *CommonName*, que deve refletir exatamente o nome de máquina. No caso deste trabalho, o mesmo certificado deve validar o nome de cada servidor da solução, assim como o nome usado pelo *UCARP*, a saber, serv_a, serv_b e domain, do domínio kirdeika.org. Isto pode ser conseguido sem o uso de *wildcard* (*) no nome de máquina, que em caso de ser usado implica em falha de segurança pois outros servidores poderiam ser introduzidos no *cluster*, de forma válida. Assim, deve ser criado um arquivo de configuração específico para o *OpenSSL* refletindo a necessidade, e para esta é usado o campo *subjectAltName* em seção própria do arquivo de configuração. Com isso, os nomes dos servidores envolvidos podem ser usados na criação de um único certificado, facilitando a manutenção do *cluster*.

Durante a instalação do *OpenSSL* é criado o arquivo /etc/ssl/openssl.cnf que contém a estrutura básica necessária. Neste arquivo, são realizadas as alterações constantes nas linhas de código a seguir, refletindo a necessidade do trabalho no *SubjectAltName*.

Na seção [req], são necessárias as seguintes alterações:

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
```

Então, na seção [req_distinguished_name], são alteradas as linhas da seção, ficando como a seguir:

```
[ req_distinguished_name ]
countryName                     = Country name ( 2 letras)
countryName_default             = BR
countryName_min                 = 2
countryName_max                 = 2

stateOrProvinceName             = State or Province Name (full name)
stateOrProvinceName_default     = DISTRITO FEDERAL

localityName                    = Locality Name (eg, city)
localityName_default            = BRASILIA

0.organizationName              = Organization Name (eg, company)
0.organizationName_default      = Pos CEUB - Mario KirdeiKa
...

organizationalUnitName          = Organizational Unit Name (eg, section)
organizationalUnitName_default  = Servidor

commonName                      = Common Name (e.g. server FQDN or YOUR name)
commonName_max                  = 64

emailAddress                    = Email Address
emailAddress_max                = 64
emailAddress_default            = kirdeikajr@gmail.com
```

Na seção [v3_req], solicitada pelo *OpenSSL* pela configuração realizada acima quando se trata de trabalho em *csr*, são alteradas as linhas ficando como a seguir:

```
[ v3_req ]

# Extensions to add to a certificate request

basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName          = @alt_names
```

Estas alterações criam a verificação da seção [alt_names], que tem o seguinte conteúdo:

```
[alt_names]
DNS.1   = serv_a.kirdeika.org
DNS.2   = serv_b.kirdeika.org
```

```
DNS.3   = domain.kirdeika.org
IP.1    = 192.168.0.10
IP.2    = 192.168.0.20
IP.3    = 192.168.0.1
```

Então, uma chave privada e a *csr* podem ser geradas:

```
# OpenSSL req -new -newkey rsa:4096 -sha256 -nodes -keyout domain.key -out domain.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country name ( 2 letras) [BR]:
State or Province Name (full name) [DISTRITO FEDERAL]:
Locality Name (eg, city) [BRASILIA]:
Organization Name (eg, company) [Pos CEUB - Mario KirdeiKa]:
Organizational Unit Name (eg, section) [Servidor]:
Common Name (e.g. server FQDN or YOUR name) []:domain.kirdeika.org
Email Address [kirdeikajr@gmail.com]:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Com a geração da *csr*, pode-se assinar o certificado com os parametros necessários:

```
# OpenSSL x509 -req -days 365 -in domain.csr -signkey POS_CEUB.key -out domain.pem -extensions
v3_req -extfile OpenSSL.cnf
Signature ok
subject=/C=BR/ST=DISTRITO FEDERAL/L=BRASILIA/O=Pos CEUB - Mario
KirdeiKa/OU=Servidor/CN=domain.kirdeika.org/emailAddress=kirdeikajr@gmail.com
Getting Private key
Enter pass phrase for POS_CEUB.key:
```

Após a inserção da senha da chave privada, o certificado foi gerado e pode ser verificado com o comando a seguir:

```
# OpenSSL x509 -text -noout -in domain.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 9291781100707772175 (0x80f309ae5e5ff70f)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=BR, ST=DISTRITO FEDERAL, L=BRASILIA, O=Pos CEUB - Mario KirdeiKa,
OU=Servidor, CN=domain.kirdeika.org/emailAddress=kirdeikajr@gmail.com
        Validity
            Not Before: Sep  1 20:27:02 2015 GMT
            Not After : Aug 31 20:27:02 2016 GMT
        Subject: C=BR, ST=DISTRITO FEDERAL, L=BRASILIA, O=Pos CEUB - Mario KirdeiKa,
OU=Servidor, CN=domain.kirdeika.org/emailAddress=kirdeikajr@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
```

```
                Public-Key: (2048 bit)
                Modulus:
                    00:bc:4b:9f:b5:28:66:60:fd:63:59:80:cd:7b:05:
                    3b:e4:ea:54:1a:26:b9:81:10:5b:f6:9c:41:20:1d:
                    1f:6e:50:7a:92:bc:d0:e8:d4:a5:90:58:31:85:71:
                    9b:28:f8:cb:fe:57:80:1c:38:46:d5:8c:66:fa:5a:
                    35:39:8d:f7:e7:20:2e:07:2e:35:04:83:89:f7:25:
                    55:cd:f3:7b:c1:76:5a:79:8c:10:cf:bf:83:6a:e1:
                    81:59:14:ad:d5:96:84:45:45:d2:ca:91:f1:82:a8:
                    66:34:45:e4:60:0e:96:1d:dd:85:27:96:19:dc:a5:
                    17:6c:8d:bd:dc:63:f8:d2:8c:ae:80:2b:62:ab:08:
                    62:e0:93:d2:17:90:b1:ad:1f:87:f4:25:44:85:41:
                    d5:ba:8e:54:40:70:fe:34:e8:80:19:1e:33:13:59:
                    8d:45:a6:63:56:72:5a:89:2f:af:b3:af:f6:d4:83:
                    06:b9:51:2e:24:e7:9d:a3:07:3b:06:45:97:60:a3:
                    f4:72:a5:29:f3:1b:ae:5e:ac:b8:5e:81:6d:26:1e:
                    ec:12:a8:89:45:b2:81:29:1f:1f:8e:da:02:ee:df:
                    56:27:d8:4e:94:1e:7a:1b:4d:0f:c2:9f:38:fc:f5:
                    1d:2a:17:9c:a8:47:67:0a:1c:db:b6:4f:24:9f:a9:
                    29:69
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Subject Alternative Name:
                DNS:serv_a.kirdeika.org, DNS:serv_b.kirdeika.org, DNS:domain.kirdeika.org, IP
Address:192.168.0.10, IP Address:192.168.0.20, IP Address:192.168.0.1
    Signature Algorithm: sha256WithRSAEncryption
         27:ac:58:d7:ac:06:04:cd:f6:99:fa:fd:c6:8e:ab:a0:5d:0e:
         33:d2:7b:d3:8e:01:19:cb:6b:e7:3d:e6:cf:7f:42:b1:2e:00:
         84:fd:70:f3:fa:43:a7:3d:9d:ae:dd:28:d0:89:85:8a:7f:03:
         bc:65:2f:c7:3c:93:0d:5c:50:93:4b:34:08:28:88:8d:a0:a5:
         e0:23:39:9e:4e:a9:f4:38:a2:79:e2:7b:f9:87:13:6c:21:b5:
         e9:bb:94:90:eb:7e:1f:8a:e0:55:d9:09:d5:ec:77:4b:1a:a4:
         d7:9c:b0:55:df:f3:ff:6b:a6:b4:d0:85:96:a7:64:c9:28:dc:
         43:6a:f7:2d:29:e1:4e:68:0b:d0:b3:fb:74:b1:06:36:45:8a:
         df:e6:35:3a:db:f8:0f:1b:61:9a:1c:27:69:b5:18:e7:12:94:
         05:44:16:38:05:12:a7:90:86:88:6b:b2:e7:c3:36:58:84:f5:
         22:b7:15:15:e8:78:cb:d0:cd:be:e5:c7:b0:f8:d0:f1:27:4f:
         20:34:91:74:e6:71:ec:64:a8:8e:2b:88:bd:e3:18:96:2d:d0:
         c6:20:a5:12:21:8c:63:ba:b3:a5:80:09:17:34:8f:de:d7:05:
         66:a1:fb:b0:d4:82:7d:16:4b:1e:72:96:07:8f:7e:9f:c1:70:
         f7:22:cb:64
```

Copia-se então os arquivos gerados para os nomes definitivos, POS_CEUB.pem para kirdeika-tcc.pem, domain.pem para serv_a.kirdeika.org e domain.key para chave_servidor.pem. Com isso, o trabalho de geração de certificados está concluído e os certificados da CA, no arquivo kirdeika-tcc.pem, e o certificado do servidor junto com sua chave privada, serv_a.kirdeika.org.pem e chave_servidor.pem, respectivamente, podem ser instalados no servidor. Deve-se atentar para o fato da necessidade de acerto das devidas permissões para o arquivo de chave privada, que neste caso está em texto plano, devido necessidade tanto do *OpenLDAP* quanto do Apache.

APÊNDICE E – *BACKUP* CRIPTOGRAFADO

Este procedimento é responsável por extrair as informações e configurações dinâmicas da base do *OpenLDAP*, criar um arquivo *tar* com a extração, adicionando ou não os *scripts* e pacotes de *software* especializados da solução conforme o horário, gerar uma chave simétrica para criptografar estes arquivos, criptografar a chave simétrica com a chave pública especial e compactar tudo, copiando para diretório local e remoto.

Em caso de qualquer falha, ou uma vez por dia, será informado o administrador, pelo *e-mail* configurado no arquivo da solução em /etc/domain.conf.

Para os procedimentos de extração de dados e configuração do *OpenLDAP* será utilizada a ferramenta *slapcat*, já disponível pela instalação do *OpenLDAP*. Importante notar que por este fato, a ferramenta *slapadd* deve ser usada quando uma restauração é necessária. Para os procedimentos de criptografia, o pacote *OpenSSL* já instalado fornece todas as ferramentas. A cópia remota será realizada pela ferramenta *scp*, do pacote *OpenSSH*, já instalado por padrão.

Para a geração de chave simétrica, é usado o executável *mkpasswd*, pertencente ao pacote *expect*. Pode ser instalado com o comando “pacman -Sy expect”.

Para o procedimento de *e-mail*, a ferramenta *ssmtp* é usada por permitir enviar *e-mails* a qualquer servidor, sem a necessidade de ser instalado um serviço de *e-mail* como *sendmail* ou *postfix* local, simplificando a solução. Para instalar o *software*, o comando “pacman -Sy ssmtp” realiza a tarefa.

A configuração do *ssmtp* é realizada em /etc/ssmtp/ssmtp.conf, e o conteúdo necessário para que as funcionalidades deste trabalho estão a seguir.

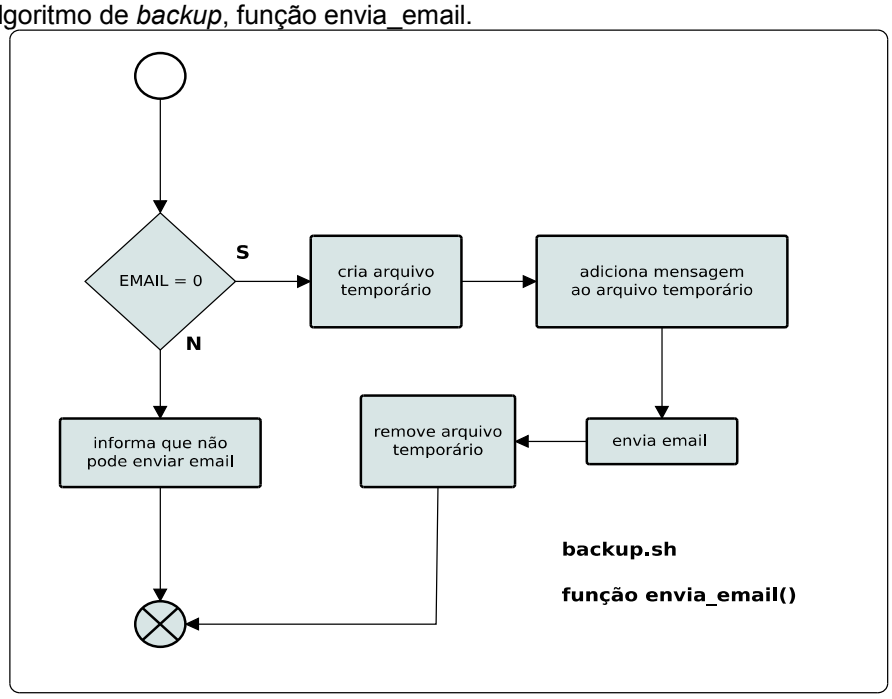
```
root=kirdeikajr@gmail.com
mailhub=smtp.gmail.com:465
rewriteDomain=
HOSTname=serv_a.kirdeika.org
UseTLS=Yes
UseSTARTTLS=Yes
AuthUser=kirdeikajr@gmail.com
AuthPass=$senha
FromLineOverride=Yes
```

Nota-se que uma conta de *e-mail* criada externamente, neste trabalho no provedor de *e-mail* Gmail, é usada, podendo ser em qualquer sistema disponível.

A chave pública será gerada externamente, a partir da chave privada, e o par será armazenado em local próprio no *HOST*, onde serão armazenados os arquivos da cópia remota. Apenas a chave pública precisa estar no servidor em que será realizado o *backup.*

O algoritmo da função de envio de *e-mail* está a seguir.

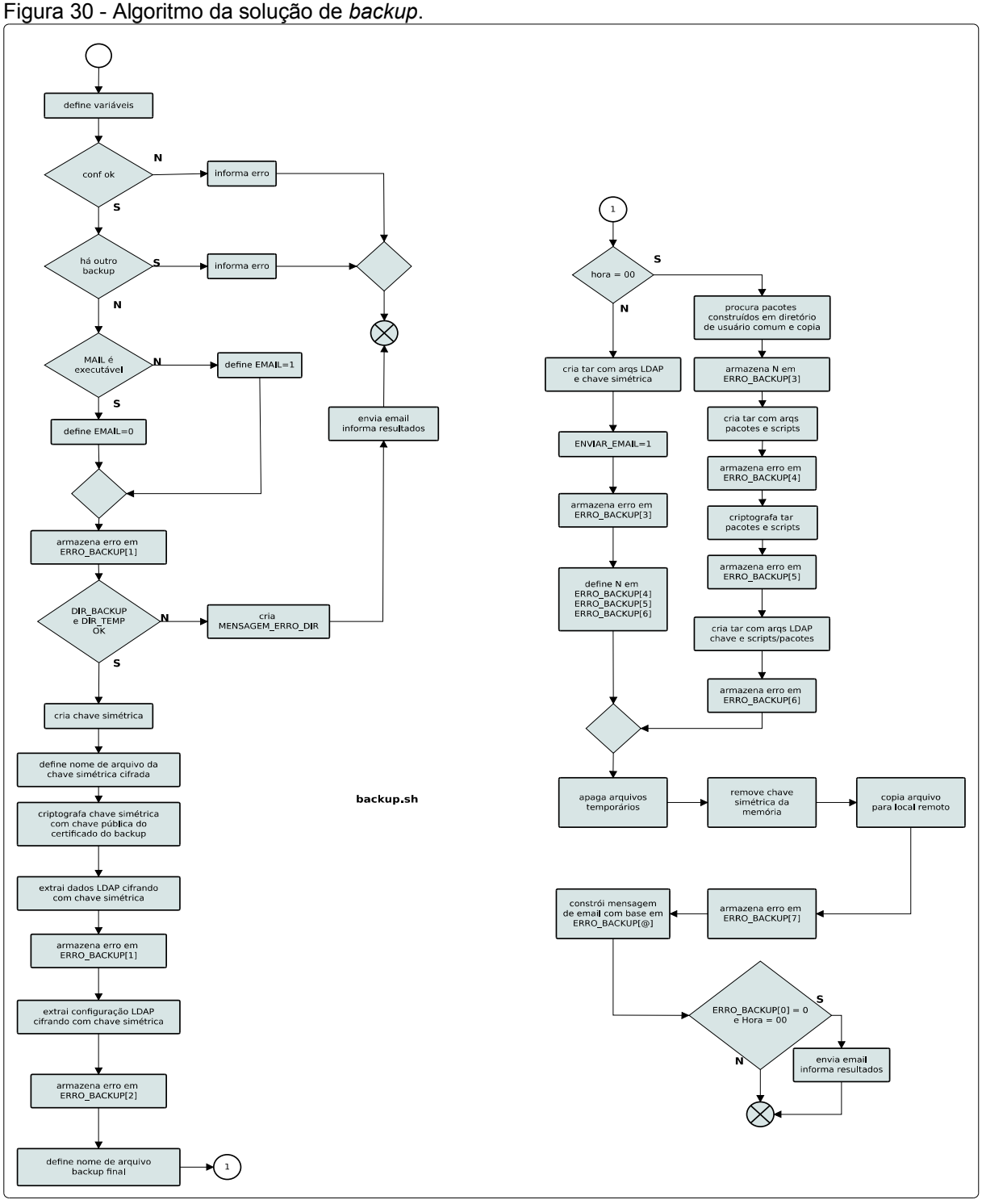
Figura 29 - Algoritmo de *backup*, função envia_email.



Fonte: Autor.

O algoritmo do procedimento de *backup* para a solução está a seguir.

Figura 30 - Algoritmo da solução de *backup*.



Fonte: Autor.

O *script* baseia-se nas funções de extração do *OpenLDAP* fornecidas pelo *slapcat* e nas funções criptográficas do *OpenSSL*.

Após a definição dos parâmetros de execução nas variáveis e verificação do arquivo principal de configuração da solução, da existência ou não de outro *backup* sendo executado, é verificada a presença do executável *mail* do pacote *ssmtp*.

Note-se que se houver outro pacote de *e-mail* instalado, como *postfix* ou *sendmail*, haverá outro executável *mail* da mesma forma e poderá ser utilizado, sem alterações no *script*.

A função *e-mail* usa este recurso, e sendo encontrado o executável *mail*, o que implica em que a variável EMAIL estará em 0, é então criado um arquivo temporário de texto como conteúdo da mensagem, que é lido e enviado ao *mail* junto com os demais parâmetros passados à função.

É executada uma verificação na condição dos diretórios temporário e de destino dos *backup's*, e em caso de problema o fato é registrado, há a tentativa de enviar *e-mail* ao administrador informando o caso e o *script* é encerrado com erro.

Se não houver problemas, é criada a chave simétrica com o comando *mkpasswd*, de tamanho configurável mas dependente do tamanho da chave pública que a cifrará. Foi obtido o valor de 256 bits em testes, considerado suficiente para a aplicação, com módulo de chave pública de 2048 bits. Este tamanho de chave pública pode ser alterado, na geração do par de chaves.

Note-se que a chave simétrica é mantida em memória, e removida assim que os arquivos necessários são cifrados. A chave é gravada em arquivo já cifrada pelo *OpenSSL*, usando a chave pública do certificado. Assim apenas com a posse da chave privada é possível decifrá-la, em tese.

Então são extraídos os dados e a configuração do *OpenLDAP*, diretamente enviados ao *OpenSSL*, compactados e já cifrados com a chave simétrica, e então gravados em disco. Com isso há economia de espaço em disco e maior segurança, não há arquivo temporário com os dados ou configuração em

texto plano no diretório temporário.

Então, a decisão a partir da hora do dia é tomada, se o *backup* será apenas dos dados e configuração, que ocorre de hora em hora, ou será completo com a inclusão dos *scripts* e pacotes de *software* construídos para a solução. No segundo caso, os *scripts* tem local definido mas os pacotes são procurados no diretório HOME do usuário comum informado e copiados para local definido.

Estes arquivos são também criptografados, e então o arquivo definitivo é criado já em seu local definitivo com todos os anteriores, conforme a decisão de hora, ou seja, se conterá ou não os *scripts* e pacotes de *software*.

São então apagados os arquivos temporários, removida a chave simétrica da memória e montada a mensagem de fim de *backup,* que é gravada em *log*, e se for possível enviar *e-mail* e o *backup* for diário, é enviado *e-mail* ao administrador.

O código resultante do algoritmo está a seguir.

```
#!/bin/bash
#
#
#Copyright (c) 2015, Mario Sergio KirdeiKa Junior
#All rights reserved.
#
#Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
#
#1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
#
#2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
#
#3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
#
#THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
#
# este script trata todo o procedimento de backup, conforme políticas estabelecidas para a solução
PIDOF=$(whereis -b pidof | awk '{print $2}')
KILL=$(whereis -b kill | awk '{print $2}')
TAR=$(whereis -b tar | awk '{print $2}')
```

```
SLAPCAT=$(whereis -b slapcat | awk '{print $2}')
MKPASSWD=$(whereis -b mkpasswd | awk '{print $2}')
OPENSSL=$(whereis -b openssl | awk '{print $2}')
MAIL=$(whereis -b mail | awk '{print $2}')
SCP=$(whereis -b scp | awk '{print $2}')
FIND=$(whereis -b find | awk '{print $2}')
LOG=/var/log/backup.log
CONF=/etc/domain.conf
CERTIFICADO_BACKUP="/root/ha/backup.pem"
DIR_SCRIPTS="/root/ha"
DIR_BACKUP="/var/backup"
DIR_TEMP="/tmp"
SERVIDOR_REMOTO="tatooine.kirdeika.org"
DIR_BACKUP_PACOTES="$DIR_BACKUP/pacotes"
USUARIO_COMUM="mario"
LOCAL_REMOTO="/home/mario/pos_ceub/TCC/backups"
USUARIO_REMOTO="mario"
declare -a ERRO_BACKUP
# verifica a configuração
if [ -f $CONF ] ; then
      . $CONF
else
      echo -ne "\n`date`\nO arquivo de configuração $CONF é inválido!\n"  >> $LOG
      exit 5
fi
if [ $($PIDOF $0 | wc -w) -ne 0 ] ; then
            echo "`date` : Existe outra instância de $0 em execução! Saindo..." >> $LOG
            exit 1
fi
if [ ! -x $MAIL ] ; then
            echo "`date` : Mail inválido, não posso enviar emails! " >> $LOG
            EMAIL=1
else
            EMAIL=0
fi
ERRO_BACKUP[0]=$EMAIL
envia_email() {
if [ $EMAIL -eq 0 ] ; then
# uso: envia_email $assunto $USER_EMAIL $EMAILADM $motivo $mensagem
TEMP=$(mktemp -p $DIR_TEMP email.XXXX)
echo -ne $4 > $TEMP
cat $TEMP | $MAIL -S "Content-Type: text/plain; charset=UTF-8" -r "$SERVIDOR <$3>" -s "$1" "Mario <$2>" 2>&1 >>
$LOG
rm -f $TEMP
else
            echo "`date` : Não consegui enviar mensagem ao $2" >> $LOG
fi
}
# se houver problema nos diretórios não posso fazer mais nada
if [ ! -d $DIR_BACKUP -o ! -d $DIR_TEMP ] ; then
MENSAGEM_ERRO_DIR="Verifique os diretórios do backup, há um problema:
Espaço em disco: `df -h`
$DIR_BACKUP :
Espaço: `du -sh $DIR_BACKUP`
`ls -l $DIR_BACKUP`
$DIR_TEMP :
Espaço: `du -sh $DIR_TEMP`
`ls -l $DIR_TEMP`
"
            echo "`date`: Falha no backup - problema em diretórios - $MENSAGEM" >> $LOG
            echo $MENSAGEM_ERRO_DIR
            envia_email "Falha no backup - problema nos diretórios" $EMAILADM $USER_EMAIL
$MENSAGEM_ERRO_DIR
            exit 2
fi
# chave simétrica
CHAVE_SIMETRICA=$($MKPASSWD -l 256)
CHAVE_SIMETRICA_CIFRADA=$DIR_TEMP/pandora_$(date "+%F-%H-%M-%S").cript.key
echo $CHAVE_SIMETRICA | $OPENSSL rsautl -encrypt -certin -inkey $CERTIFICADO_BACKUP -out
$CHAVE_SIMETRICA_CIFRADA
# se tudo ok, extraio LDAP
FILE_LDAP_DADOS="$DIR_TEMP/ldap_dados_$(echo $SERVIDOR | cut -d. -f1)_$(date "+%F-%H-%M-%S").ldif.cript"
```

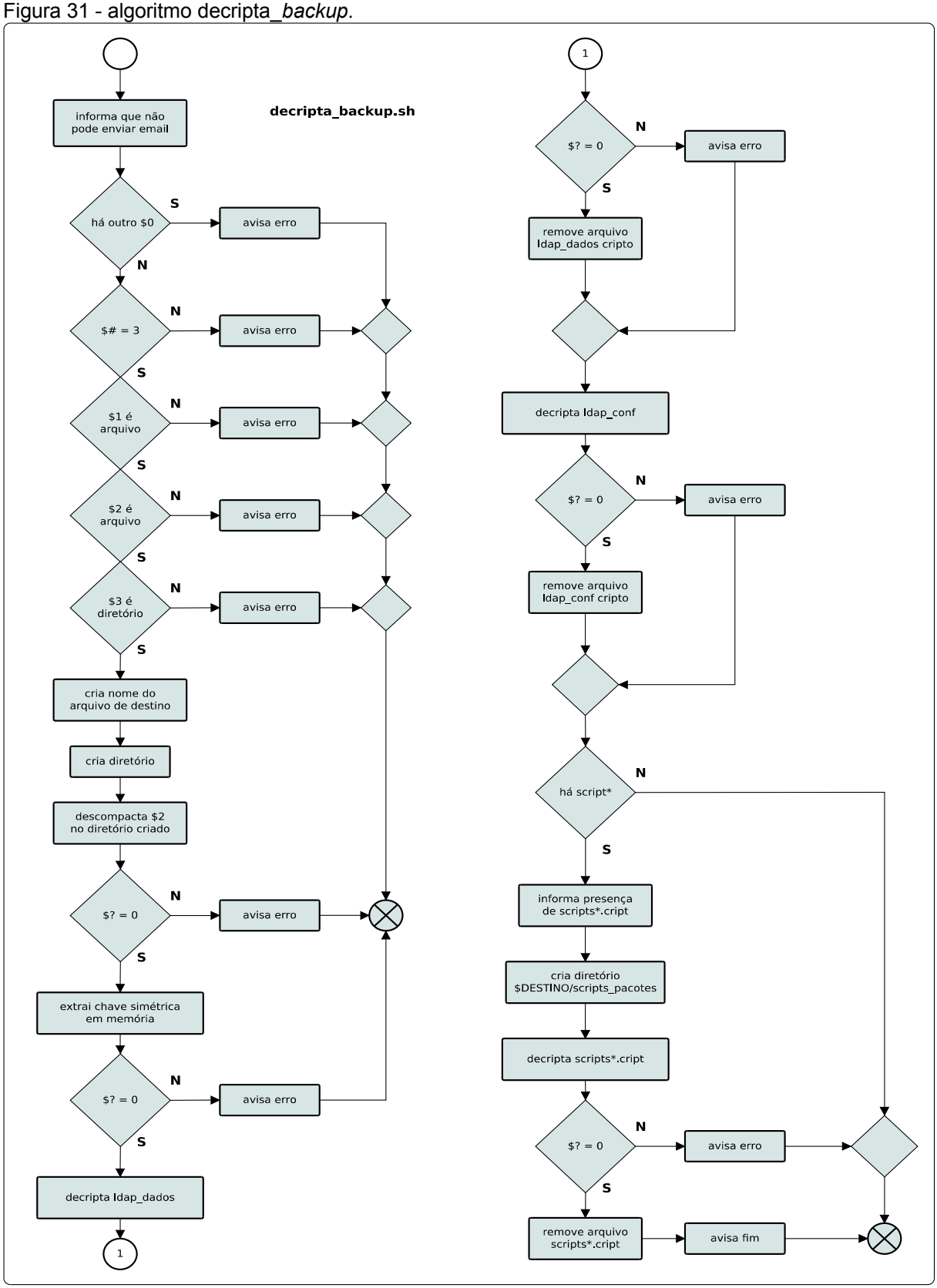
```
FILE_LDAP_CONF="$DIR_TEMP/ldap_conf_$(echo $SERVIDOR | cut -d. -f1)_$(date "+%F-%H-%M-%S").ldif.cript"
$SLAPCAT 2>/dev/null | $OPENSSL aes-256-cbc -e -z -k ${CHAVE_SIMETRICA} -out $FILE_LDAP_DADOS 2>&1 >>
$LOG
ERRO_BACKUP[1]=$?
$SLAPCAT -n 0  2>/dev/null | $OPENSSL aes-256-cbc -e -z -k $CHAVE_SIMETRICA -out $FILE_LDAP_CONF 2>&1 >>
$LOG
ERRO_BACKUP[2]=$?
# definindo arquivo final de backup
ARQUIVO_FINAL="$DIR_BACKUP/backup.$(echo $SERVIDOR | cut -d. -f1).$(date "+%F-%H-%M-%S").tar.gz"
if [ $(date "+%H") -ne 00 ] ; then
        ENVIAR_EMAIL=1
        $TAR -cf - $CHAVE_SIMETRICA_CIFRADA $FILE_LDAP_DADOS $FILE_LDAP_CONF 2>/dev/null | gzip >
$ARQUIVO_FINAL 2>> $LOG
        ERRO_BACKUP[3]=$?
        ERRO_BACKUP[4]=N
        ERRO_BACKUP[5]=N
        ERRO_BACKUP[6]=N
else
        # mantendo cópia dos pacotes construídos em /var/backup/pacotes
        $FIND $(eval echo ~$USUARIO_COMUM) -iname *.pkg.tar.xz -exec cp -f {} $DIR_BACKUP_PACOTES/ \; 2>&1
>> $LOG
        ERRO_BACKUP[3]=N
        SCRIPTS=$DIR_TEMP/scripts_$SERVIDOR_$(date "+%F-%H-%M-%S").tar.gz
        $TAR -cf - /etc/openldap /root/ha $DIR_BACKUP_PACOTES 2>/dev/null | gzip > $SCRIPTS
        ERRO_BACKUP[4]=$?
        $OPENSSL aes-256-cbc -e -z -k $CHAVE_SIMETRICA -in $SCRIPTS -out $SCRIPTS.cript 2>&1 >> $LOG
        ERRO_BACKUP[5]=$?
        $TAR -cf - $CHAVE_SIMETRICA_CIFRADA $FILE_LDAP_DADOS $FILE_LDAP_CONF $SCRIPTS.cript
2>/dev/null | gzip > $ARQUIVO_FINAL 2>> $LOG
        ERRO_BACKUP[6]=$?
        ENVIAR_EMAIL=0
fi
# apagando temporários
for i in "$FILE_LDAP_DADOS $FILE_LDAP_CONF $CHAVE_SIMETRICA_CIFRADA $SCRIPTS $SCRIPTS.cript" ; do
        rm -f $i 2>/dev/null
done
unset CHAVE_SIMETRICA
# copiando para local remoto
$SCP $ARQUIVO_FINAL $USUARIO_REMOTO@$SERVIDOR_REMOTO:$LOCAL_REMOTO/ 2>&1 >> $LOG
ERRO_BACKUP[7]=$?
# mensagem de finalização
MENSAGEM=$(echo "\nFinalização do BACKUP de $(date)\n\nResultados\n\n")
for i in `seq 0 1 7` ; do
        case $i in
                0)if [ ${ERRO_BACKUP[0]} -eq 0 ] ; then MENSAGEM+="Envio de EMAIL habilitado\n" ; else
MENSAGEM+="Envio de EMAIL desabilitado\n" ; fi ;;
                1)if [ ${ERRO_BACKUP[1]} -eq 0 ] ; then MENSAGEM+="Extração de dados do LDAP ok\n" ; else
MENSAGEM+="Falha na extração de dados do LDAP\n" ; FALHA=1 ; fi ;;
                2)if [ ${ERRO_BACKUP[2]} -eq 0 ] ; then MENSAGEM+="Extração de configuração do LDAP ok\n" ;
else MENSAGEM+="Falha na extração de configuração do LDAP\n" ; FALHA=1 ; fi ;;
                3)if [ ${ERRO_BACKUP[3]} -eq 0 2>/dev/null ] ; then MENSAGEM+="Realizado backup simples.\n" ;
else MENSAGEM+="Realizado backup diário\n" ; fi ;;
                4)if [ ${ERRO_BACKUP[4]} -eq 0 2>/dev/null ] ; then MENSAGEM+="Scripts e pacotes salvos ok\n" ;
elif [ ${ERRO_BACKUP[4]} == "N" ] ; then MENSAGEM+="Backup simples, não são salvos Scripts e pacotes\n" ; else
MENSAGEM+="Falha na criação de arquivo de scripts e pacotes\n" ; FALHA=1 ;fi ;;
                5)if [ ${ERRO_BACKUP[5]} -eq 0 2>/dev/null ] ; then MENSAGEM+="Scripts e pacotes
criptografados ok\n" ; elif [ ${ERRO_BACKUP[5]} == "N" ] ; then MENSAGEM+="Backup simples, não são salvos
Scripts e pacotes\n" ; else MENSAGEM+="Falha na encriptação do arquivo de scripts e pacotes\n" ; FALHA=1 ;fi ;;
                6)if [ ${ERRO_BACKUP[6]} -eq 0 2>/dev/null ] ; then MENSAGEM+="Arquivo final diário ok\n"  ; elif
[ ${ERRO_BACKUP[6]} == "N" ] ; then MENSAGEM+="Backup simples, não são salvos Scripts e pacotes\n" ; else
MENSAGEM+="Falha na criação de arquivo final\n" ; FALHA=1 ;fi ;;
                7)if [ ${ERRO_BACKUP[7]} -eq 0 ] ; then MENSAGEM+="Cópia do arquivo final ok\n" ; else
MENSAGEM+="Falha na cópia remota do arquivo final\n" ; fi ;;
        esac
done
echo -ne $MENSAGEM >> $LOG
echo "Erros = ${ERRO_BACKUP[@]}" >> $LOG
if [ ${ERRO_BACKUP[0]} -eq 0 -a $(date "+%H") -eq 00 ] ; then
        envia_email "Relatório de backup" $EMAILADM $USER_EMAIL "$MENSAGEM"
fi
exit 0
```

Para auxílio no momento da necessidade de decriptação de algum *backup*, foi desenvolvido o algoritmo *decripta_backup*. O *script* resultante recebe como parâmetros o caminho completo da chave privada, do arquivo a ser decriptado e o diretório base onde será realizada a operação.

É criado um subdiretório no local informado, e todos os arquivos são extraídos e decriptados, com o auxílio da chave privada informada. Um relatório ao final da execução do *script* informa sobre os eventos e localização de arquivos.

O algoritmo a seguir descreve as ações, realizadas ao inverso do *script* de *backup*.

Figura 31 - algoritmo decripta_*backup*.



Fonte: Autor.

O código resultante está a seguir:

```
#!/bin/bash
#
#
#Copyright (c) 2015, Mario Sergio KirdeiKa Junior
#All rights reserved.
#
#Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
#
#1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
#
#2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
#
#3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.
#
#THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
#
# este script trata todo o procedimento de backup, conforme políticas estabelecidas para a solução
PIDOF=$(whereis -b pidof | awk '{print $2}')
KILL=$(whereis -b kill | awk '{print $2}')
TAR=$(whereis -b tar | awk '{print $2}')
OPENSSL=$(whereis -b openssl | awk '{print $2}')
LOG=/var/log/backup.log
CONF=/etc/domain.conf
DIR_TEMP="/tmp"
if [ $($PIDOF $0 | wc -w) -ne 0 ] ; then
        echo "`date` : Existe outra instância de $0 em execução! Saindo..." >> $LOG
        exit 1
fi
if [ $# -ne 3 ] ; then
        echo "Uso: $0 chave_privada arquivo_backup destino_extração"
        exit 1
elif [ ! -e $1 ] ; then
        echo "Chave privada inválida!"
        exit 1
elif [ ! -e $2 ] ; then
        echo "Aquivo de backup não encontrado!"
        exit 1
elif [ ! -d $3 ] ; then
        echo "Diretório de destino da extração inválido!"
        exit 1
fi
DESTINO="$3/extração_backup/$(echo $2 | cut -d. -f2)_$(echo $2 | cut -d. -f3 )"
echo -ne "\n`date` - Iniciando decriptação de $2 em $3 usando a chave privada $1\n\n"
mkdir -p $DESTINO
$TAR xzf $2 -C $DESTINO/ --strip-components=1 2>/dev/null
if [ $? -ne 0 ] ; then
        echo "Houve erro no $TAR!!!! Sinto..."
        exit 2
fi
CHAVE_SIMETRICA="$($OPENSSL rsautl -decrypt -inkey $1 -in $(ls $DESTINO/pandora* 2>/dev/null))"
if [ $( echo $CHAVE_SIMETRICA | wc -c) -lt 255 ] ; then
        echo "Não decriptei a chave, saindo"
        exit 2
```

```
fi
$OPENSSL aes-256-cbc -d -z -k $CHAVE_SIMETRICA -in $(ls $DESTINO/ldap_dados*.cript) -out $(ls
$DESTINO/ldap_dados* | sed 's/.cript//') 2>/dev/null
if [ $? -ne 0 ] ; then
        echo "Falha na decriptação do arquivo de dados do OpenLDAP..."
else
        echo "Arquivo de dados $(ls $DESTINO/ldap_dados*.cript) decriptado com sucesso."
        rm -f $(ls $DESTINO/ldap_dados*.cript) 2>/dev/null
fi
$OPENSSL aes-256-cbc -d -z -k $CHAVE_SIMETRICA -in $(ls $DESTINO/ldap_conf*.cript) -out $(ls
$DESTINO/ldap_conf* | sed 's/.cript//') 2>/dev/null
if [ $? -ne 0 ] ; then
        echo "Falha na decriptação do arquivo de configuração do OpenLDAP..."
else
        echo "Arquivo de configuração $(ls $DESTINO/ldap_conf*.cript) decriptado com sucesso."
        rm -f $(ls $DESTINO/ldap_conf*.cript) 2>/dev/null
fi
if [ $(ls $DESTINO/scripts_2*.cript 2>/dev/null | grep "cript" | wc -l ) -gt 0 ] ; then
        echo "Detectei a presença do arquivo de scripts e pacotes...."
        mkdir -p $DESTINO/scripts_pacotes
        $OPENSSL aes-256-cbc -d -z -k $CHAVE_SIMETRICA -in $(ls $DESTINO/scripts*.cript) | tar -xzf - -C
$DESTINO/scripts_pacotes/
        if [ $? -ne 0 ] ; then
                echo "Houve erro na decriptação/extração do arquivo de scripts e pacotes"
        else
                echo "Decriptação/extração do arquivo de scripts e pacotes ok."
                rm -f $(ls $DESTINO/scripts*.cript)
        fi
fi
echo -ne "\n\n`date` - Fim da decriptação / extração, veja resultados em $DESTINO\n"
```