Universidade Federal de Santa Maria — UFSM

Dissertação de Mestrado

UM SISTEMA DE GERENCIAMENTO INTEGRADO E AUTOMATIZADO DE SERVIÇOS E COMPUTADORES DE REDES LOCAIS

Diego Luís Kreutz

Programa de Pós-Graduação em Engenharia de Produção — PPGEP

Santa Maria, RS, Brasil

2005

UM SISTEMA DE GERENCIAMENTO INTEGRADO E AUTOMATIZADO DE SERVIÇOS E COMPUTADORES DE REDES LOCAIS

por

Diego Luís Kreutz

Dissertação apresentada ao Curso de
Mestrado do Programa de Pós-Graduação em
Engenharia da Produção, área de concentração
em Tecnologia da Informação, da Universidade
Federal de Santa Maria (UFSM, RS), como
requisito parcial para obtenção do grau de
Mestre em Engenharia da Produção

PPGEP

Santa Maria, RS, Brasil

2005

Universidade Federal de Santa Maria
Centro de Tecnologia
PPGEP

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

UM SISTEMA DE GERENCIAMENTO INTEGRADO E AUTOMATIZADO DE SERVIÇOS E COMPUTADORES DE REDES LOCAIS

elaborada por
Diego Luís Kreutz

como requisito parcial para obtenção do grau de
Mestre em Engenharia da Produção

COMISSÃO EXAMINADORA:

Prof. Dr. Benhur de Oliveira Stein
(Presidente/Orientador — Departamento de Eletrônica e Computação — UFSM)

Prof. Dr. Lisandro Zambenedetti Granville
(Departamento de Computação Aplicada — UFRGS)

Prof. Dr. Antonio Augusto Medeiros Fröhlich
(Departamento de Ciência da Computação — UFSC)

Santa Maria, 30 de março de 2005

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Kreutz, Diego Luís

Um Sistema de Gerenciamento Integrado e Automatizado de Serviços e Computadores de Redes Locais / Diego Luís Kreutz. – Santa Maria: PPGEP, 2005.

128 f.: il.

Dissertação (mestrado) – Universidade Federal de Santa Maria. PPGEP, Santa Maria, BR–RS, 2005. Orientador: Benhur de Oliveira Stein.

1. Redes de computadores. 2. Gerenciamento integrado. 3. Automatização. 4. Escalabilidade. 5. Tolerância a falhas. 6. Componentes independentes. 7. Gerenciamento centralizado. 8. Distribuição. I. Stein, Benhur de Oliveira. II. Título.

A confecção desta dissertação foi realizada através do uso
de recursos de processamento de textos da ferramenta LaTeX 2_ε e do editor *Vim*

"Any intelligent fool can make things
bigger, more complex, and more violent.
It takes a touch of genius – and a lot
of courage – to move in the opposite direction."
(Albert Einstein)

"The average mind is easily content
with inherited and acquired things,
or with the dicta of parents and teachers,
because it is much easier to imitate than to create."
(Emma Goldman)

"Two things are infinite: the universe
and human stupidity; and I'm not sure
about the universe."
(Albert Einstein)

"The first step toward wisdom is
mastering the obvious."
(David Johnson)

"Tell me and I forget.
Teach me and I remember.
Involve me and I learn."
(Benjamin Franklin)

À você leitor, que (talvez) terá a paciência para aturar as mais
de 50 páginas de corpo e outras 50 de anexos. ;-)
Se é para dedicar a alguém, este alguém só pode ser a sociedade. ;-)

AGRADECIMENTOS

A todos que de alguma forma contribuíram para a realização deste trabalho.

Sem esquecer, agradecemos também ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro e concessão de bolsas de pesquisa e desenvolvimento ao Laboratório de Sistemas de Computação (LSC) da UFSM.

RESUMO

Gerenciamento de redes de computadores é uma área que exige competência, criatividade, imaginação, ação, observação, autonomia e muito trabalho. Apesar de anos de investimentos realizados nessa área, ainda continua sendo uma área com carência de ferramentas e profissionais qualificados para o gerenciamento dos mais diversos domínios administrativos. Neste contexto, este trabalho descreve o projeto e desenvolvimento de um sistema integrado, dinâmico e automatizado capaz de simplificar e agilizar várias tarefas do dia-a-dia de administradores de redes.O sistema se propõe também a ser uma forma de integração entre ferramentas de gerenciamento existentes, tanto estendendo seus recursos quanto suas funcionalidades. Os resultados demonstram a validade e aplicabilidade do sistema. Foi desenvolvido um protótipo do sistema, que já pôde ser aplicado em redes de computadores para a automatização de várias tarefas administrativas.

Palavras-chave: Redes de computadores, gerenciamento integrado, automatização, escalabilidade, tolerância a falhas, componentes independentes, gerenciamento centralizado, distribuição.

An Integrated and Automated Network Management System for Services and Computers of Local Networks

ABSTRACT

The management of computer networks is an area that demands competence, creativeness, imagination, action, observation, autonomy and a lot of work. Even today it is still an area with a lack of tools and professionals qualified to the management of the most diverse administrative domains. In this context, this work describes the project and development of an integrated, dynamic and automated system that is able to simplify and reduce the time required by for many daily tasks of network administrators. The proposed system can also be used as a way of integrationg existing management tools, making use of their resources and functionalities. The results validate and show the appliance of the system. A prototype of the system was developed, that could already be applied to automate many administrative tasks of computer networks.

Keywords: computer networks, integrated management, automation, scalability, fault tolerance, independent components, centralized management, distribution.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

AIX	Advanced Interactive Executive (versão Unix da IBM)
API	Application Program Interface
ASCII	American Standard Code for Information Interchange
Cfengine	Host Configuration Engine
CGI	Common Gateway Interface
CMIP	Management Information Protocol
CORBA	Common Object Request Broker Architecture
DCAST	Doshisha Cluster Auto Setup Tool
DIT	Directory Information Tree
DNS	Domain Name Service
FAI	Fully Automatic Installation
FTP	File Transfer Protocol
GNU	GNU's Not Unix
HTML	Hypertext Markup Language
HTTP	HyperText Transfer Protocol
idaNMOS	Integrated, Dynamic and Automated Network Management and Operation System
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
INMS	Integrated Network Management System
IPS	Intrusion Prevention System
IP	Internet Protocol
ISO	International Organization for Standardization

JDBC	Java DataBase Connectivity
JNDI	Java Naming and Directory Interface
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MIB	Management Information Base
MIT	Management Information Tree
MP3	format for compressing digital audio files
NetConf	Network Configuration Protocol
NetInfo	Network Information System
NIS	Network Information Service
NMC	Network Management Center
OSCAR	Open Source Cluster Application Resources
OSI	Open Systems Interconnection
PDA	Personal Digital Assistant
Perl	Practical Extraction and Reporting Language
PHP	Hypertext Preprocessor
PIKT	Problem Informant/Killer Tool
RPC	Remote Procedure Call
SDK	Software Development Kit
SMIT	System Management Interface Tool
SNMP	Simple Network Management Protocol
SNM	SunNet Manager
SOAP	Simple Object Access Protocol
SSL	Secured Sockets Layer
SUE	Standardized Unix Environment
TLS	Transport Layer Security
WWW	World Wide Web
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

O gerenciamento de redes de computadores é um assunto de pesquisa e desenvolvimento relativamente antigo, com publicações na área desde aproximadamente 1970. No entanto, ainda é uma área que carece de pesquisa, desenvolvimento de soluções e capacitação profissional (HOEPERS, 2004). O mercado de redes é grande e a oferta de mão de obra qualificada não é o suficiente para atender a demanda.

Essas carências são ainda mais graves quando se leva em consideração a crescente importância que as redes de computadores vem tendo nos mais variados ramos da atividade humana. Há pouco tempo teve início uma verdadeira explosão de conectividade. Mais e mais redes estão sendo instaladas e precisam ser adequadamente gerenciadas. Gerenciar redes heterogêneas, onde a qualquer momento podem surgir novos serviços ou dispositivos de rede que precisam ser controlados e mantidos, é uma verdadeiro desafio. Para enfrentar esse desafio, é importante que os administradores dessas redes possam ser apoiados por ferramentas abrangentes de gerenciamento de dispositivos e serviços de redes de computadores, que automatizem o maior número possível de tarefas administrativas.

Um indicativo da carência de soluções gerenciais é a falta de ferramentas de gerenciamento de redes [1] abrangentes, largamente difundidas e aceitas pelos administradores de redes. Grande parte das ferramentas existentes solucionam apenas problemas isolados, não tendo uma abrangência significativa sobre o gerenciamento, configuração, atualização e monitoramento de sistemas e dispositivos de rede. Uma forma de amenizar o problema do isolamento e independência entre as diversas soluções é fornecer algum mecanismo (ou sistema) que seja capaz de facilmente incorporar novas funcionalidades e que ao mesmo tempo possibilite controlar e fazer uso das soluções existentes. As vantagens dessa integração vão desde a redução nas possibilidades de falhas, através do uso de processos automatizados de verificação e manutenção das informações de configuração, até a agilização e simplificação de tarefas do dia-a-dia de administradores de redes.

Neste contexto, este trabalho tem como propósito o projeto e desenvolvimento de um sistema integrado de gerenciamento para redes locais (denominado de `idaNMOS` [2]). O objetivo

[1] No contexto em questão, ferramentas de gerenciamento de redes são tratadas como sendo aquelas destinadas ao controle e manutenção das configurações dos sistemas distribuídos pela rede.

[2] **I**ntegrated, **D**ynamic and **A**utomated **N**etwork **M**anagement and **O**peration **S**ystem - porém, no decorrer do

do sistema é controlar de forma integrada, automática e dinâmica serviços e computadores de redes locais, simplificando o trabalho dos administradores destas redes. Após o sistema ter sido implantado, atualizações de configuração de serviços e computadores poderão ser efetuadas, de forma automática, com eficiência e simplicidade, apenas alimentando a base de dados do sistema com informações pertinentes ao funcionamento e configuração dos serviços e computadores da rede.

Outros aspectos importantes da proposta deste trabalho são a escalabilidade, flexibilidade, independência de plataforma e linguagem de programação, auto-manutenção e tolerância a falhas. Estas características muitas vezes não estão presentes ou são pouco desenvolvidas em soluções de gerenciamento existentes. Praticamente nenhuma ferramenta disponível engloba todas essas características.

A próxima seção pormenoriza os objetivos de trabalho. Na seção seguinte é apresentada a organização geral do texto.

1.1 Objetivos do trabalho

Os objetivos do trabalho podem ser discriminados em:

Projetar um sistema integrado de gerenciamento de redes locais. Através da análise de necessidades administrativas de redes locais, identificar funcionalidades e recursos necessários para se automatizar o gerenciamento de uma rede de computadores e projetar uma arquitetura que suporte o desenvolvimento de um sistema com essas características.

Desenvolver o sistema `idaNMOS`. O desenvolvimento do sistema tem como propósito mostrar a viabilidade e aplicabilidade da arquitetura proposta para o gerenciamento integrado e automatizado de serviços e computadores de redes locais.

Validar o sistema. A validação servirá para mostrar com o protótipo desenvolvido algumas características, aplicabilidades e funcionalidades do `ida`. Os testes serão realizados na rede do Núcleo de Ciência da Computação (NCC). Características como escalabilidade, consistência, segurança, automatização e flexibilidade serão avaliados. Com isso espera-se comprovar a eficácia e utilização do sistema para casos reais de gerenciamento de redes.

1.2 Organização do Texto

O texto é dividido em 6 capítulos:

Capítulo 2: aborda alguns conceitos de administração de redes, a tecnologia de agentes inteligentes e características esperadas em sistemas de gerenciamento de redes. Neste capítulo são agrupados e apresentados também os principais trabalhos relacionados ao contexto do `ida`.

texto será referenciado apenas como `ida`

Capítulo 3: apresenta, inicialmente de forma abstrata, a arquitetura e características do sistema proposto.

Capítulo 4: aborda detalhes e especificidades da implementação do sistema. Este capítulo apresenta também uma visão gráfica da arquitetura implementada.

Capítulo 5: apresenta e comenta dados estatísticos e características que validam e comprovam a aplicabilidade e funcionalidade do sistema.

Capítulo 6: finalizando, esta última parte do trabalho apresenta algumas inferências sobre o `ida` e o conteúdo presente nos demais capítulos. Além disso, relaciona algumas idéias e propostas de trabalhos futuros.

Informações adicionais a algumas seções e capítulos estão distribuídas pelos apêndices em anexo:

Apêndice A: apresenta maiores detalhes sobre cada uma das ferramentas comentadas no capítulo 2 e descreve os detalhes de algumas das características desejadas em sistemas de gerenciamento de redes de computadores.

Apêndice B: contém informações adicionais sobre a configuração e arquitetura da base dados e exemplos reais de entradas cadastradas na base de informações do `ida`.

Apêndice C: este apêndice apresenta maiores detalhes sobre as ferramentas utilizadas para o desenvolvimento do `ida`. São também descritas as linguagens de programação suportadas para o melhoramento, adaptação ou extensão do sistema.

Apêndice D: descreve em maiores detalhes os componentes e recursos implementados e provê um exemplo prático de código.

2 GERENCIAMENTO DE REDES DE COMPUTADORES

Este capítulo contém uma base bibliográfica para a proposta de gerenciamento desenvolvida e validada neste trabalho. São apresentadas as áreas funcionais, a tecnologia de agentes e os trabalhos relacionados.

Com o objetivo de diminuir a complexidade e criar especialidades do gerenciamento de redes, a ISO (International Organization for Standardization) juntamente com a OSI (Open Systems Interconnection), resolveu dividir o gerenciamento de redes em áreas funcionais. A seção 2.1 apresenta rapidamente essas divisões.

A seção 2.2 apresenta algumas características e formas de aplicação de agentes no contexto de gerenciamento de redes. A tecnologia de agentes é alvo de pesquisa e aplicação crescente no gerenciamento de redes de computadores.

Os trabalhos relacionados são agrupados e comentados na seção 2.3.6. No final da desta seção são abordadas também as principais características esperadas em sistemas integrados de gerenciamento de redes. Uma descrição mais detalhada dos trabalhos relacionados encontra-se no apêndice A.

2.1 Áreas Funcionais

A ISO/OSI a fim de amenizar a complexidade do gerenciamento de redes criou cinco divisões básicas: gerência de falhas, gerência de configuração, gerência de desempenho, gerência de contabilização e gerência de segurança (LEINWAND; FANG, 1995). Cada uma destas áreas atua sobre fatores que determinam o bom funcionamento do conjunto de componentes de uma rede.

Gerência de Desempenho: o monitoramento dos dispositivos de rede é importante a fim de avaliar o comportamento individual e coletivo dos componentes que fazem parte da rede. Os dados resultantes do monitoramento podem servir de base para o controle de qualidade de serviço e geração e análise de estatísticas de comportamento da rede. Estes dados empíricos podem ser utilizados para evitar falhas como sobrecarga de tráfego e quebra de prestação de serviços;

Gerência de Falhas: o objetivo desta área é detectar e corrigir problemas na rede. Para isso, as

ferramentas de gerenciamento devem ser capazes de encontrar, isolar e identificar as possíveis e conhecidas falhas de componentes físicos e lógicos da rede. A partir da correlação e averiguação dos dados coletados medidas de correção podem ser tomadas;

Gerência de Contabilização: armazena e disponibiliza dados de utilização dos recursos da rede. O objetivo destes dados é formar uma base para a organização, manutenção e planejamento da estrutura de distribuição e funcionamento dos dispositivos da rede. As aplicações vão desde redução de custos até a previsão de capacidade e análise de uso;

Gerência de Configuração: é destinada a simplificar tarefas como a instalação, manutenção e ativação de sistemas. Além disso, informações sobre estrutura, localização, detalhes do dispositivo e administrador responsável são alguns dados úteis para manter a organização da rede e, por conseqüência, são informações que fazem parte do contexto de gerencia de configuração;

Gerência de Segurança: compreende os diversos mecanismos que mantém a integridade da rede. Isso envolve o controle de acesso através de sistemas de autorização e autenticação, monitoramento de atividades críticas ou suspeitas, identificação de possíveis falhas de acesso, prevenção contra tentativas maliciosas de corrupção de programas e protocolos em geral, entre outras coisas. Medidas preventivas englobam o uso de criptografia, cópias de segurança, *firewalls* e monitoramento de processos e sistemas de arquivos.

O presente trabalho situa-se na divisão de gerência de configuração. O objetivo é ter um sistema capaz de gerenciar configurações de dispositivos e serviços de rede em geral e, inclusive, de outros sistemas de gerenciamento.

2.2 Agentes

Um agente é qualquer coisa que é capaz de perceber o ambiente e tomar ações sobre este ambiente através de eventos, em propósito de sua própria agenda, podendo levar a novos comportamentos futuros (FRANKLIN; GRAESSER, 1996).

Os agentes normalmente são classificados através de aspectos como: estruturas de controle, ambientes, linguagens ou aplicações. Propriedades como autonomicidade, orientação a objetivo, temporalidade, continuidade, comunicatividade, aprendizado, mobilidade, flexibilidade e atribuições são características igualmente inerentes a agentes (FRANKLIN; GRAESSER, 1996; FRANCESCHI; ROISENBERG; BARRETO, 2002). Subconjuntos dessas propriedades formam agentes dos mais variados tipos.

Em gerência de rede é comum o uso de agentes passivos. Estes são conhecidos por não utilizarem técnicas de IA e não possuírem autonomia (FRANCESCHI; ROISENBERG; BARRETO, 2002). No entanto, atualmente estão cada vez mais sendo utilizados e pesquisados, no âmbito de gerência de redes, agentes autônomos, agentes móveis e agentes inteligentes (KONA;

XU, 2002; MULLER, 1997; PILS, 2001; PUJOLLE et al., 2001; PUJOLLE; RUBINSTEIN; DUARTE, 2001; RUBINSTEIN; DUARTE; PUJOLLE, 2002). Os gentes móveis, por exemplo, no contexto de gerenciamento de redes são considerados uma área de pesquisa promissora (BIESZCZAD; WHITE; PAGUREK, 1998).

2.3 Sistemas de Gerenciamento

Existe um grande número de ferramentas para o gerenciamento de sistemas em redes de computadores. Para facilitar a caracterização e exemplificação dos recursos atualmente providos por vários protocolos e ferramentas, eles foram divididos em quatro classes básicas:

Protocolos: são padrões de comunicação e controle utilizados para gerenciar dispositivos de rede;

Linguagens de configuração: envolvem plataformas de configuração e gerenciamento que tenham uma linguagem própria e fixa para a manipulação dos dados de configuração de aplicativos da rede;

Ferramentas de gerenciamento distribuído: são aquelas que podem simultaneamente gerenciar vários dispositivos ou sistemas de uma rede;

Ferramentas de gerenciamento de uma única máquina: são aquelas destinadas ao controle de uma única máquina ou dispositivo de rede.

Existem algumas ferramentas que não se encaixam nestas quatro classes. Estas são sistemas de aplicação específica e restrita, se levado em conta o contexto abordado neste trabalho.

As seções seguintes relacionam ferramentas e ambientes que fazem parte das quatro classificações básicas. A seção 2.3.6 relaciona algumas das características desejáveis em sistemas de gerenciamento de redes.

2.3.1 Protocolos

Os protocolos de gerenciamento de dispositivos e sistemas de rede são recursos padronizados que permitem o controle e operação de uma vasta gama de componentes, dos mais diversos fabricantes. Uma das metas dos protocolos de gerenciamento é justamente definir padrões de operação que podem ser utilizados e implementados em *hardware* ou *software* por diferentes fabricantes de equipamentos de rede.

O SNMP (Simple Network Management Protocol - Protocolo Simples para o Gerenciamento de Rede) (FRYE et al., 2000) é um dos protocolos mais conhecido e utilizados pela comunidade acadêmica e por organizações e ambientes corporativos. Como o próprio nome sugere, a arquitetura e a forma de aplicação deste protocolo são bastante simples e podem ser implementadas em *hardware*. Esses são alguns dos motivos que levaram a sua ampla aceitação e uso no mercado de redes de computadores. Apesar disso, até sua versão atual, este protocolo apresenta

algumas deficiências e problemas que motivaram, inclusive, o desenvolvimento de novos procotolos e ferramentas de gerenciamento como o NETCONF (ENNS, 2004). Entre as maiores deficiências do SNMP estão a segurança e a dificuldade de aplicação em alguns ambientes complexos (com numerosos e/ou diversos sistemas e componentes de rede) e que exigem suportes gerenciais mais dinâmicos e flexíveis.

Maiores informações sobre estes protocolos de gerenciamento podem ser vistas no apêndice A (seção A.1).

2.3.2 Linguagens de Configuração

As linguagens de configuração compreendem sistemas de configuração e gerenciamento que possuem um padrão de descrição de dados próprio e definido. Estes sistemas são normalmente baseados em representações com níveis de abstração maiores que linguagens interpretadas como *Shell Scripts*, Perl e Python, ou seguem os padrões como o XML (eXtensible Markup Language).

Entre as ferramentas de descrição de configuração de sistemas de rede podem ser citadas: PAN (CONS; POZNANSKI, 2002), PIKT (OSTERLUND, 2000), Cfengine (BURGESS, 1995), LCFG (ANDERSON, 1994), Alchemist (DUNNAVANT, 2000) e Arusha (Arusha Project Team, 2003; HOLGATE; PARTAIN, 2001). Cada uma possui suas características, ambientes alvos e particularidades (maiores detalhes podem ser vistos no apêndice A - seção A.2).

Um dos limitantes de praticamente todos esses sistemas são as linguagens próprias e definidas. Ao utilizar a ferramenta o usuário fica praticamente restrito à linguagem que a acompanha. Além disso, poucos desses sistemas usam uma única base de informações, centralizada e integrada. Outros recursos como a relação de dependências de dados, automatização de gerenciamento, flexibilidade, granularidade administrativa e atualização dinâmica e automática de sistemas são alguns dos pontos fracos dessas ferramentas.

2.3.3 Ferramentas de Gerenciamento Distribuído

Ferramentas de gerenciamento distribuído permitem, a partir de um único ponto, controlar e manter sistemas e computadores de uma rede. Estes sistemas partem do pressuposto que existe uma base de informações (ou conjuntos de arquivos ou dados que descrevem informações de configuração de sistemas) que permitirá a atualização e manipulação, de modo centralizado, das configurações de serviços e máquinas de uma rede de computadores.

NetInfo (Apple Computer, Inc., 1998, 1999), Config4gnu (LONG; YACKOSKI, 2002), ManageWise (FORD; ELLIOT, 1996; Novell, Inc., 1999a), SunNet Manager (SunSoft, 1995a), SUE (CERN / IT / PDP group, 2000), NetView (HOCHSTETLER et al., 2000), OpenView (Hewlett-Packard Development Company, L.P., 2004; Dell Inc., 1999) e FreeNMS (Grupo de Gerência de Redes - Laboratório Metropoa / PUCRS, 2004) são exemplos de sistemas de gerenciamento distribuído de dispositivos e serviços de rede. Muitas destas ferramentas são destinadas a ambientes proprietários e particulares, não sendo aplicáveis ao gerenciamento geral de

redes de computadores com sistemas e dispositivos diversificados.

O NetInfo é apenas um protocolo com uma base de dados centralizada que torna possível a manutenção das informações da rede em um único local, acessível por todos os dispositivos e sistemas da rede. Para isso, cada serviço ou aplicação terá que ter uma interface capaz acessar e compreender o formato de dados do NetInfo. Não obstante, ele não é um sistema geral de configuração e manutenção de sistemas e dispositivos de rede.

Sistemas como o Config4gnu, ManageWise, SunNet Manager, SUE, NetView e OpenView são destinados a ambientes específicos e possuem funcionalidades particulares para o gerenciamento desses ambientes. Permitem o controle e monitoramento de dispositivos de rede e alguns sistemas. No entanto, não proporcionam um ambiente integrado, dinâmico e automático para o gerenciamento dos mais diversos serviços, sistemas e componentes de uma rede de computadores.

O FreeNMS é basicamente uma ferramenta de monitoramento e diagnóstico de redes de computadores. Apesar disso, disponibiliza também recursos para o controle de dispositivos remotos através do protoloco SNMP. Este, por sua vez, é um dos limitantes para um gerenciamento geral e flexível de dispositivos e sistemas (ver seção A.1.1 do apêndice A).

A seção A.3 do apêndice A contém descrições mais detalhadas sobre as ferramentas de gerenciamento distribuído comentadas nesta seção.

2.3.4 Ferramentas de Gerenciamento de uma Única Máquina

Os sistemas de gerenciamento de uma única máquina têm como objetivo prover recursos para agilizar e simplificar a administração de um único computador. Eles são dotados de interfaces e funcionalidades que permitem um gerenciamento de serviços e aplicações de uma determinada máquina. Algumas destas ferramentas possuem interfaces Web que possibilitam um gerenciamento remoto do computador.

O SMIT(SEGURA, 2000), Webmin(SCHMIDT, 1999), Linuxconf(GÉLINA, 2003) e as CLIs(LEE; CHOI, 2002) são exemplos de sistemas destinados ao gerenciamento de computadores e sistemas locais. Cada um possui seu ambiente alvo e suas especificidades. Mas, em essência, servem para o mesmo fim.

Ferramentas de gerenciamento de uma única máquina não são uma boa opção para redes de médio e grande porte, onde dezenas, centenas ou até milhares de computadores e dispositivos de redes precisam ser gerenciados. Mesmo para redes de pequeno porte, envolvendo alguns poucos computadores, essas ferramentas de gerenciamento de uma única máquina podem não ser uma boa opção. A aplicação delas no âmbito de uma rede resultaria em uma administração mais onerosa e menos tolerante a falhas devido a falta de integração e centralização gerencial.

A seção A.4 (apêndice A) aborda com maiores detalhes cada uma dessas ferramentas.

2.3.5 Outras Ferramentas de Uso Específico e Limitado

Além das ferramentas citadas nas quatro classificações das seções anteriores, existem outras ferramentas destinadas a ambientes e funções específicas ou ainda desenvolvidas para casos particulares de gerenciamento. Logo, elas não são aplicáveis a qualquer domínio administrativo ou situação gerencial. Entre estas ferramentas podem ser citadas: Sun Management Center, SNMPMonitor, PSWeM e NetCrunch. Maiores detalhes e referências podem ser encontrados na seção A.5.2 do apêndice A.

Existem também vários conjuntos de ferramentas e ambientes destinados a contextos de aplicação restrita. Entre os mais conhecidos por administradores de domínios estão os sistemas de instalação e gerenciamento de aglomerados de computadores. Estes normalmente são compostos por máquinas com ferramentas e sistemas homogêneos.

O objetivo principal de ambientes e ferramentas como DCAST, FAI, Update-Cluster, OSCAR, Rocks e Ka Clustering Tools é prover mecanismos que tornam eficiente e fácil a instalação de conjuntos idênticos de máquinas que irão cooperar entre si para resolver problemas complexos da computação. Os ambientes e ferramentas fornecem recursos que agilizam a criação e implantação de aglomerados de computadores por usuários, ou administradores, com pouca ou nenhuma experiência na área em questão.

A seção A.5, do apêndice A, apresenta algumas características e aplicações das ferramentas (citadas anteriormente) que podem ser utilizadas na administração de aglomerados de computadores. Referências sobre cada uma delas também são fornecidas no apêndice.

2.3.6 Características Desejáveis

Os sistemas de gerenciamento de redes são ferramentas destinadas ao controle e manutenção de máquinas e serviços distribuídos pelo domínio administrativo. As áreas de aplicação vão desde o monitoramento de dispositivos até a configuração geral de programas e computadores.

No contexto de gerenciamento integrado, foram identificadas algumas características que são capazes de prover boas soluções gerenciais. Entre as características consideradas essenciais para soluções integradas de gerenciamento podem ser incluídas: *uma única fonte de informação*, que provê todos os dados necessários para gerenciar dispositivos e serviços de rede, evitando redundâncias de dados; *manutenção da atual arquitetura dos sistemas*, não forçando o desenvolvimento de novas versões ou a modificação dos sistemas existentes; *solução baseada em componentes*, possibilitando o rápido e fácil desenvolvimento de novos componentes e adaptação de necessidades incomuns de diferentes domínios administrativos; *possibilidade de execução distribuída*, aumentando a abrangência de gerenciamento para sistemas distribuídos ou mesmo para distribuir os dados e a carga de computação através dos recursos computacionais disponíveis; *fácil utilização*, aonde o administrador precisa, geralmente, apenas manter os dados na fonte de informação, que estes serão automaticamente considerados pelos sistemas afetados; e *tolerância a falhas*, habilitando a detecção e correção automática de problemas na disponibilidade de serviços da rede, por exemplo.

A seguir são apresentadas outras características (e algumas das abordadas no parágrafo anterior são estendidas/reforçadas), que incorporam maiores funcionalidade e abrangem uma maior variedade de casos de aplicação de um sistema de gerenciamento.

Escalabilidade: um sistema de gerenciamento de redes deve ser capaz de gerenciar redes de tamanhos diversos (FUJISAKI; HAMADA; KAGEYAMA, 1997).

Gerenciamento de atualização de *software*: esta é uma tarefa importante e necessária em uma rede de computadores. É comum fabricantes, desenvolvedores e usuários lançarem pacotes de atualização para falhas de funcionamento e falhas de segurança. Estes últimos pacotes, em especial, devem ser monitorados e observados por administradores de rede. Além disso, algumas máquinas, em determinados momentos, estarão inacessíveis (CONS; POZNANSKI, 2002) e por isso as atualizações devem ser processadas de modo assíncrono. As atualizações devem ainda ser computadas em momentos adequados, evitando a parada de sistemas vitais (SVENTEK, 2001).

Validação dos dados de configuração: em uma base de dados única pode ser fácil cometer erros que comprometerão muitos sistemas (CONS; POZNANSKI, 2002). Logo, a validação dos dados constantes na base de dados é imprescindível. Essa validação pode ser a verificação da presença e coerência de relações e dependências, a existência e a configuração de máquinas importantes.

Flexibilidade/customização: uma ferramenta de administração de redes deveria ser customizável para a organização ou o indivíduo que a está utilizando (FUJISAKI; HAMADA; KAGEYAMA, 1997). A própria variedade de sistemas existentes (ver seção 2.3) na área de gerenciamento de redes é um bom indicativo da necessidade de flexibilidade e liberdade. Administradores de rede gostam de projetar suas soluções e utilizar as linguagens que mais lhe convêm. Atualmente até agentes móveis estão sendo utilizados para implementar características de flexibilidade no gerenciamento de rede (PULIAFITO; TOMARCHIO, 2002). O pressuposto do uso de agentes é o gerenciamento distribuído, possibilitando mobilidade de código, uso de código sob demanda e migração de tarefas que exigem processamento para nós com menores cargas de processamento.

Extensibilidade: um sistema de gerenciamento deve ser extensível ao ponto de abranger a maior parte dos dispositivos e serviços de uma rede. A adaptação de novos componentes ou recursos deve ser possível, sempre buscando oferecer um bom grau de liberdade aos administradores de domínios. No desenvolvimento de componentes de um sistema deveria ser permitido o uso de qualquer linguagem (LONG, 2002).

Praticidade: após implantado, o sistema deve ser prático e útil para o gerenciamento dos mais diversos sistemas de uma rede de computadores. Atualizações, instalações e migrações de componentes e serviços devem ser possíveis e de fácil realização (LONG, 2002).

Tolerância a Falhas: um sistema de gerenciamento de redes deve ser tolerante a falhas (FUJISAKI; HAMADA; KAGEYAMA, 1997; MURDOCH, 2003). No caso de imprevistos ou anomalias ocorrerem na rede, o sistema de gerenciamento deve prover recursos que sejam o suficiente para mantê-lo funcionando, mesmo em ambientes adversos. Disponibilizar instrumentos para prever, evitar e corrigir falhas de serviços e máquinas da rede também é algo desejável em uma ferramenta de gerenciamento. Recursos desse gênero amenizam problemas não-previstos e facilitam o controle e a organização da rede.

Segurança: é um ponto importante em todos os níveis de gerenciamento e operação de redes. Desde o controle em tempo real de pacotes através de fronteiras de segurança e coalizão (SHYNE; MARKLE, 2000) até o gerenciamento de acesso e autenticação em nível de sistema e aplicação. Outro aspecto importante são as políticas gerais de segurança, onde são necessários detalhadas interpretação e análise por parte dos administradores de sistemas (MOFFETT; SLOMAN, 1993). A segurança em relação ao controle de acesso ao repositório de dados de uma ferramenta de administração de redes deve estabelecer quem pode fazer o quê, sobre quais informações. Isso tanto em nível de componentes do sistema quanto em nível de usuários administrativos. Definir as propriedades e prioridades de acesso é essencial em sistemas que serão gerenciados por mais de uma pessoa (LONG, 2002).

Automatização: é um recurso importante para garantir confiabilidade e precisão na manutenção de informações de configuração de sistemas (ANDERSON; SCOBIE, 2002). A automatização visa evitar erros ou esquecimentos comuns em tarefas manuais. Quando o número de sistemas gerenciados cresce é imprescindível um bom grau de automatização. Processos de atualização de dados de aplicativos e máquinas devem ser acionados de forma transparente e automática.

Integração: a possibilidade de integrar o controle do maior número possível de dispositivos e sistemas é algo desejável em um sistema de gerenciamento (KIM; CHO, 2000; LEE, 2001). Relações de dependências entre dados de configuração e informações gerais da base de dados devem ser possíveis. Neste caso, a atualização ou alteração de um dado levará à detecção automática dos dependentes (que mantém algum vinculo ou indireção aos dados em questão) e as atualização de dispositivos e serviços serão realizadas de forma transparente e automática. Além disso, a base de dados deve suportar diferentes tipos e formatos de dados a fim de garantir um maior grau de integração possível.

Simplicidade: simplicidade de projeto e implementação é importante (HOLGATE; PARTAIN, 2001). Um sistema complexo pode facilmente levar a erros e falhas difíceis de serem detectadas. Logo, um sistema composto por vários componentes independentes é uma opção a ser cogitada no desenvolvimento de sistemas de gerenciamento complexos e abrangentes. Componentes simples, que no conjunto conseguem realizar tarefas complexas, é uma

das estratégias de engenharia de *software* mais adequadas para a construção de sistemas distribuídos ou ferramentas de médio e grande porte.

Uniformidade: disponibilizar e manter métodos de acesso padrão aos dados e ferramentas de gerenciamento do sistema pode aumentar a usabilidade e facilidade de manutenção e extensão. Não forçar os usuários a utilizarem métodos obscuros e limitados. Permitir sim um uso, extensão e acesso uniforme aos dados e recursos do sistema.

Administração remota: possibilitar aos usuários do sistema que tarefas corriqueiras de manutenção e controle possam ser feitas remotamente, de preferência através de métodos simples e seguros.

Registro de ações: registrar o que foi alterado no sistema é algo importante e útil para possíveis e futuras recuperações de estados de configuração. Logo, quanto mais ricos forem os registros de modificações realizadas por administradores ou componentes do sistema melhores serão as chances de operações de correção ou retorno a um estado consistente do sistema terem sucesso (LONG, 2002).

Distribuição: arquiteturas de gerenciamento de redes no modelo cliente-servidor podem não proporcionar ambientes adequados para redes de médio e grande porte. Mesmo em redes de pequeno porte com algumas dezenas de serviços e dispositivos de rede podem não ser comportadas por sistemas de gerenciamento centralizados. Sistemas distribuídos de administração e operação de redes são uma opção emergente (FUJISAKI; HAMADA; KAGEYAMA, 1997; PULIAFITO; TOMARCHIO, 2002; ADHICANDRA; PATTINSON; SHAGHOUCI, 2003). No desenvolvimento de ferramentas desse gênero estão sendo utilizados cada vez mais agentes móveis (PULIAFITO; TOMARCHIO, 2002; ADHICANDRA; PATTINSON; SHAGHOUCI, 2003). No entanto, esta tecnologia ainda está em plena fase de desenvolvimento e evolução, carecendo de aspectos como segurança e confiabilidade. Mas, a abordagem utilizada usando agentes móveis é desejável e necessária em muitos domínios administrativos. Recursos para distribuir os dados e fornecer mobilidade de código, distribuindo cargas de processamento, são exemplos de aplicações que viriam a agregar escalabilidade e flexibilidade a sistemas de gerenciamento de domínios administrativos.

Abstração: ao invés de arquivos de configuração, representações estruturadas e mais completas são uma boa opção para o gerenciamento de muitas máquinas e sistemas (CONS; POZNANSKI, 2002). Muitas vezes é bastante custoso chegar-se a um bom nível de abstração. Mas, o gasto de esforços nesse sentido vale a pena principalmente quando muitas máquinas e sistemas serão gerenciados de forma integrada e automatizada através da abstração de dados. Além disso, quanto mais dados forem colocados na abstração, mais úteis eles se tornarão (CONS; POZNANSKI, 2002).

Ferramentas textuais devem ser o suficiente: na administração de sistemas nem sempre é possível usar plataformas ou interfaces gráfica elaboradas para realizar as mais diversas tarefas do dia-a-dia de gerentes de sistemas. Ferramentas textuais simples devem ser o suficiente para resolver os mais diversos tipos de problemas em gerenciamento avançado de redes de computadores (HOLGATE; PARTAIN, 2001).

Monitoramento: o monitoramento de serviços, sistemas e da própria ferramenta de gerenciamento é fundamental para determinar o estado atual da rede e detectar eventos emitidos por sistemas (CASSAR et al., 1996). Esses dados são úteis para diagnosticar e identificar problemas e aprimorar o funcionamento da rede.

Políticas: são basicamente objetivos estabelecidos sobre objetos-alvo (MOFFETT; SLOMAN, 1993; MOFFETT; SLOMAN; TWIDLE, 1990). Quando um objetivo é atingido normalmente uma ação é realizada sobre o objeto alvo. Políticas podem ser expressas como conjuntos de regras de autorização de acesso ou medidas a serem tomadas quando um cenário atingir uma determinada configuração. Em sistemas de gerenciamento de redes as políticas são utilizadas como instrumentos para diagnosticar problemas, resolver impasses, acionar recursos de recuperação ou correção de falhas, definir regras e prioridades de utilização de recursos, acionar notificações de alerta, delimitação de domínios de acesso, entre outras aplicações.

Granularidade administrativa: é importante possibilitar que múltiplos administradores atuem simultaneamente sobre pontos específicos (distintos ou iguais) da base de dados e dos componentes do sistema. Além disso, a restrição de acesso a conteúdos específicos a cada componente do sistema também torna possível um maior controle sobre o uso e acesso aos dados do sistema, garantindo integridade, confiabilidade e segurança dos dados.

2.4 Resumo

Este capítulo apresentou algumas tecnologias e ferramentas existentes que podem auxiliar no desenvolvimento de soluções gerenciais e na administração de redes de computadores. Foram citadas também as principais características desejáveis em sistemas integrados e abrangentes de gerenciamento de dispositivos e sistemas de rede.

Nesse contexto, o objetivo do `ida` é suprir as características essenciais e o maior número possível das demais características (ver seção 2.3.6). A meta básica do sistema é proporcionar simplicidade de manutenção de sistemas, automatização e integração gerencial.

O apêndice A contém dados mais detalhados das ferramentas abordadas neste capítulo. Informações sobre funcionamento, aplicação, características, objetivos particulares e abrangência dos sistemas podem ser encontradas nesse apêndice.

O próximo capítulo apresenta a proposta do sistema `ida`, que tem como objetivo suprir algumas dessas características, deficientes em outras ferramentas.

3 ARQUITETURA DO SISTEMA PROPOSTO

Este capítulo tem como objetivo apresentar a arquitetura geral e as principais características e funcionalidades do `ida`, a solução de gerenciamento integrado de redes de computadores proposta neste trabalho. Para tanto, a próxima seção apresenta a arquitetura geral do sistema. As seções seguintes apresentam maiores detalhes do sistema. Por fim, são comentadas as principais características e vantagens do `ida`.

Informações complementares quanto a dados de configuração da base de dados podem ser encontradas no apêndice B.

3.1 Arquitetura Geral

Esta seção apresenta a arquitetura geral do sistema. Inicialmente é apresentada a arquitetura básica do sistema. Na seqüência são abordados alguns dos componentes do sistema.

3.1.1 Arquitetura Básica

O `ida` pretende disponibilizar ao usuário final uma arquitetura simples, extensível, dinâmica e segura para o gerenciamento integrado de redes de computadores. A premissa é dar liberdade e opções de escolha ao administrador do sistema. Desta forma, ele poderá continuar utilizando muitas de suas ferramentas e táticas de gerenciamento de serviços, computadores e dispositivos de rede conjuntamente ao `ida`.

A figura 3.1 apresenta uma visão resumida do sistema. Através dela pode-se perceber que o `ida` é composto basicamente por uma base de dados, agentes e módulos. Cada um destes componentes tem as seguintes funcionalidades básicas:

Base de dados: Mantém informações atualizadas dos dispositivos e serviços da rede. Esses dados servirão de entrada aos módulos e agentes do sistema.

Módulos: atualizam os arquivos de configuração e causam a execução de determinados serviços de acordo com as descrições da base de dados.

Agentes: controlam a execução de componentes do sistema de acordo com as configurações da base de dados. Alguns tipos de agentes possuem níveis maiores de permissões e podem

atualizar, verificar a consistência e modificar as informações da base de dados.

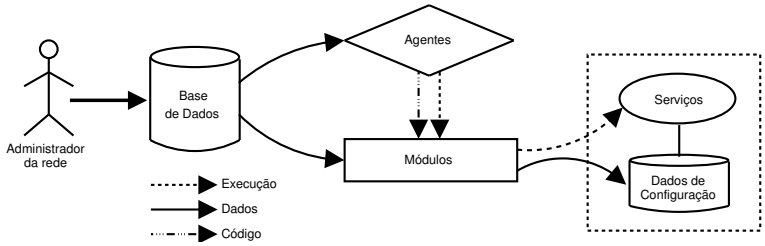

Figura 3.1: Visão resumida do sistema

Um exemplo simples de funcionamento e aplicação dos módulos e agentes é ser o controle e atualização do serviço de DNS. O módulo de manutenção do DNS gera as tabelas de resolução de nomes no formato esperado pelo servidor DNS a partir dos dados na base de dados e reinicializa o serviço para que as novas configurações tenham efeito. O agente correspondente causa a execução do módulo sempre que detectar alterações na base de dados referentes ao DNS.

Existem diversas categorias de módulos e agentes. Cada categoria possui algumas características e funcionalidades específicas. Na próximas seções as diferentes categorias de componentes do sistema serão abordadas com maiores detalhes.

A figura 3.2 ilustra a arquitetura geral do sistema. Ela é composta por módulos de geração e atualização de configurações de máquinas e serviços, agentes de serviços que controlam a execução dos módulos e agentes de monitoramento.

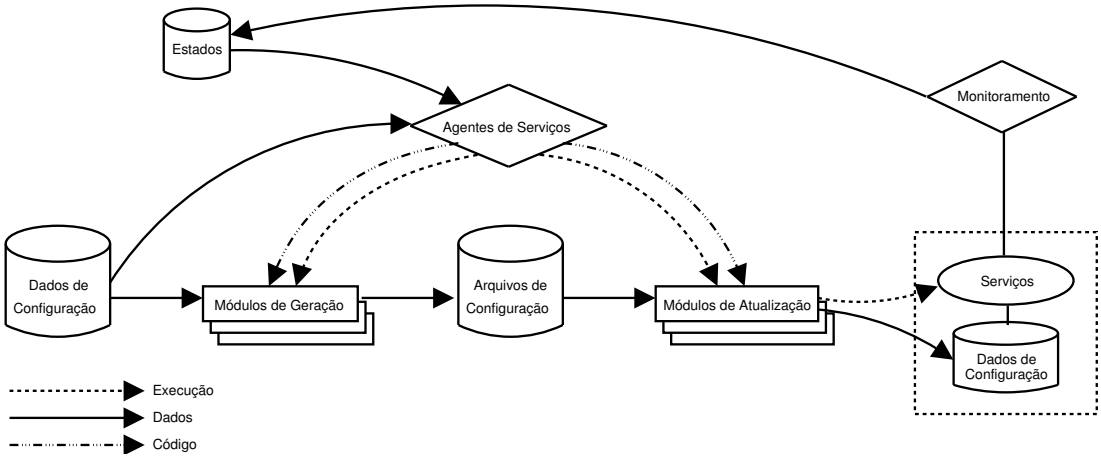

Figura 3.2: Arquitetura geral do sistema

Os estados de monitoramento e das informações da base de dados servem como ponto de referência aos agentes de serviços. Estes estados determinarão a necessidade ou não de execução de algum módulo de geração ou atualização.

O propósito básico do sistema é o gerenciamento da configuração de máquinas e serviços de uma rede.

As próximas seções apresentam as diferentes categorias de módulos e agentes do sistema. Alguns detalhes de funcionamento e aplicação são igualmente abordados.

3.1.2 Módulos

As funcionalidades básicas destes componentes são gerar e copiar configurações necessárias para manter atualizados serviços e máquinas da rede. Os módulos são componentes simples (e sem nenhuma inteligência) que apenas processam dados ou executam comandos quando acionados. Sendo assim, é fácil incluir novos módulos de geração e atualização para novos serviços ou máquinas.

Existem três tipos de módulos:

Geração: geram dados de configuração, a partir das informações da base de dados, para máquinas e serviços, no formato esperado por estes;

Atualização: copiam dados de configuração, da base de dados para o sistema de arquivos local, e executam os comandos necessários para a atualização de uma máquina ou serviço;

Execução: executam seqüências de comandos. São classificados como interfaces de execução geral de comandos.

A figura 3.3 apresenta uma visão simplificada dos módulos do sistema. Nela pode ser observado que estes componentes são simples e apenas geram algum resultado a partir de informações e configurações da base de dados. Esta característica permite um controle distribuído de máquinas e serviços da rede a partir de um único ponto de entrada, a base de dados.

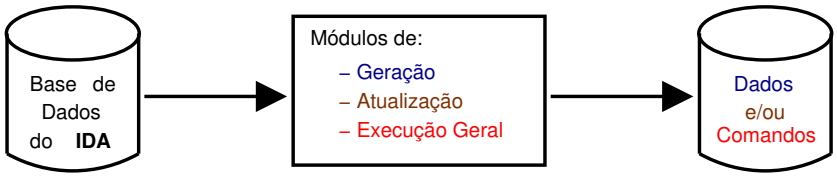

Figura 3.3: Módulos em uma visão simplificada

3.1.2.1 Módulos de Atualização

Eles são responsáveis pela configuração e atualização de serviços e sistemas em geral. Entre os módulos do sistema, os módulos de atualização (MAs) são os mais simples e fáceis de serem desenvolvidos ou estendidos.

A figura 3.4 apresenta uma visão geral desses componentes do `ida`. Os dados e comandos são extraídos da base de dados. A partir destas informações serviços, máquinas ou sistemas [1] são atualizados. Esse processo pode ser uma simples cópia de um arquivo de configuração de um serviço e sua re-inicialização (para que as atualizações tenham efeito) ou a execução de múltiplos comandos para a configuração, atualização, instalação, remoção ou ativação de serviços, máquinas ou sistemas.

[1] Neste contexto, quando em conjunto com serviços e máquinas, o termo sistemas refere-se a todos os programas, equipamentos e aplicações em geral que não se enquadram na denominação de serviço ou máquina. Um serviço é um programa essencial para o funcionamento e uso da rede. São considerados serviços programas de DNS, DHCP, NFS, Samba, SMTP, SSH, entre outros que são necessários para o uso e funcionamento da rede.

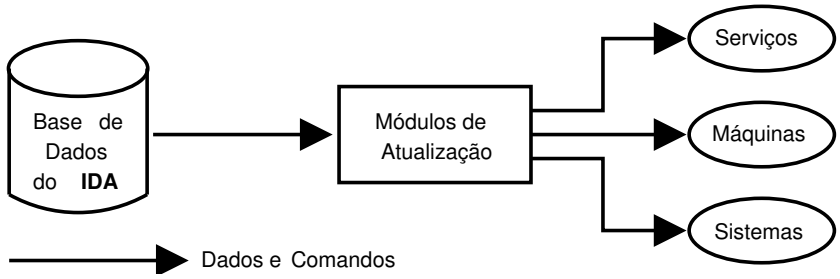

Figura 3.4: Visão geral da aplicação dos módulos de atualização

3.1.2.2 Módulos de Geração

Um módulo de geração gera os arquivos de configuração para um determinado serviço ou máquina da rede. Os dados necessários para a geração dos arquivos de configuração são extraídos da base de dados. Cada serviço, máquina ou sistema poderá ter um módulo de geração. No entanto, este componente não é necessário caso as informações contidas na base de dados estejam em um formato adequado aos arquivos de configuração de sistemas. Neste caso um módulo de atualização seria o suficiente.

A figura 3.5 representa o funcionamento básico dos módulos de configuração. A partir das informações da base de dados são gerados os dados formatados para a configuração (ou reconfiguração) de sistemas.

Figura 3.5: Módulos de configuração

Os dados de configuração gerados por um módulo de geração serão posteriormente usados por um módulo de atualização. Estes dados estão formatados e prontos para serem copiados aos locais de configuração de serviços ou máquinas.

3.1.2.3 Módulos de Execução Geral

Os módulos de execução geral servem de interface para a execução remota e distribuída de comandos. A figura 3.6 ilustra o funcionamento desses componentes.

Um módulo de execução recebe uma lista de comandos da base de dados e executa os comando-a-comando. Caso a execução tenha sido bem sucedida, a lista atual é marcada como processada. Neste caso, em uma próxima ativação do módulo serão apenas executados os novos comandos, ainda não processados.

A utilidade destes componentes vai desde a execução distribuída e simultânea de comandos para atualização geral de sistemas até a realização de tarefas emergenciais de recuperação de funcionalidades de uma máquina. Um caso de aplicação seria quando o sistema não está acei-

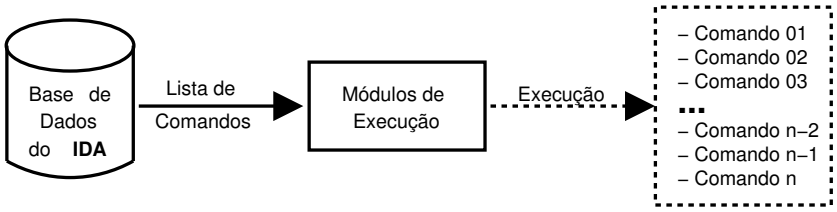

Figura 3.6: Módulos de execução geral

tando a autenticação de usuários e comandos remotos são lançados através de um módulo de execução geral para tentar re-estabelecer a autenticação da máquina.

3.1.3 Agentes

As funções dos agentes [2]. variam desde o controle e execução de módulos do sistema até o monitoramento de máquinas e serviços. Devido a isto, constituem componentes vitais à execução do sistema como um todo.

3.1.3.1 Agentes de Serviços

A função básica de um agente de serviços é controlar a execução dos módulos de geração e atualização. Em cada máquina da rede haverá ao menos um agente de serviço.

A figura 3.7 apresenta uma visão abstrata de agentes de serviços. Eles executam os agentes de geração e atualização quando necessário, ou seja, quando alterações relevantes aos módulos tenham ocorrido na base de dados.

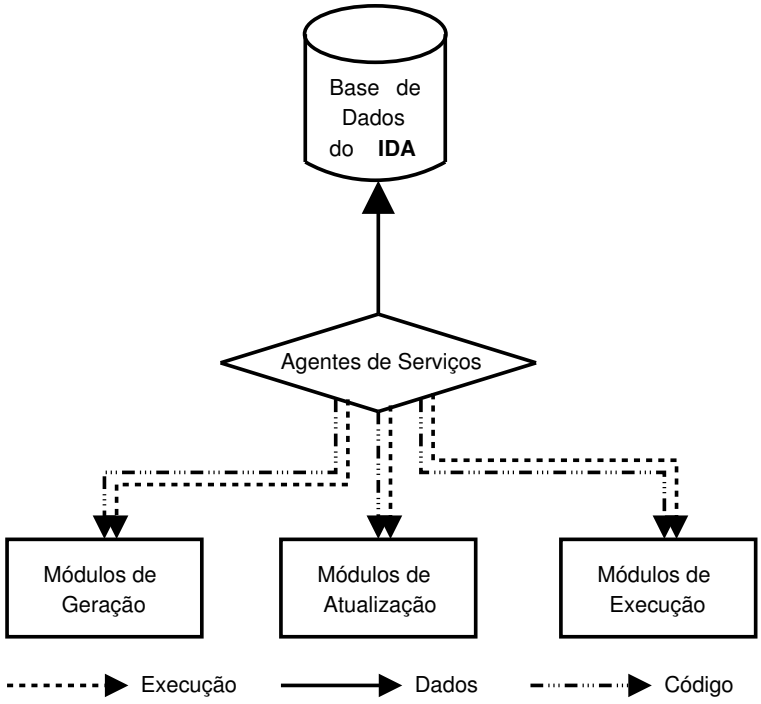

Figura 3.7: Agentes de Serviços

[2]Estes agentes não tem nenhuma relação com os agentes SNMP.

As setas de código da figura representam a atualização de código do componente, quando necessária. Este controle automático e dinâmico da atualização de código fornece mecanismos extras de manutenção do sistema propriamente dito.

3.1.3.2 Agentes de Monitoramento

A função básica dos agentes de monitoramento é analisar o estado de máquinas e serviços da rede e identificar falhas na prestação de serviços e computadores não acessíveis. A figura 3.8 apresenta uma visão geral do funcionamento destes agentes. Os dados de monitoramento dos diferentes serviços, máquinas e dispositivos são reportados à base de dados.

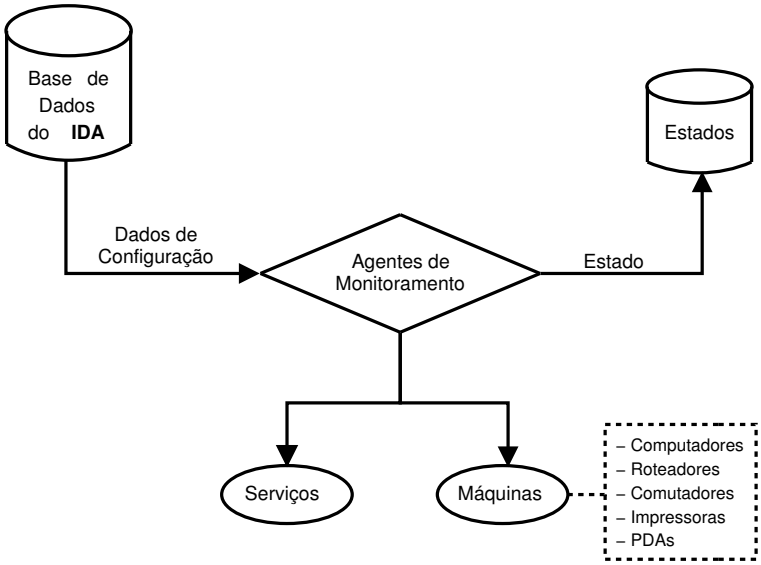

Figura 3.8: Agentes de Monitoramento

3.2 Complementos Arquiteturais

Esta seção tem como objetivo apresentar maiores detalhes sobre a arquitetura do sistema. É utilizada uma abordagem *top down*. Além disso, novos componentes são apresentados. Estes, no geral, são utilizados para resolver problemas de segurança ou para simplificar alguns dos componentes da arquitetura anteriormente apresentada.

A figura 3.9 apresenta uma visão geral da arquitetura do sistema. Constam como novos componentes os agentes locais e os agentes de controle. A função dos agentes de controle é realizar tarefas vitais ao sistema. Logo, são considerados componentes altamente seguros e confiáveis. Os agentes locais, por sua vez, vêm para simplificar a manutenção e gerenciamento dos agentes de serviços.

Os agentes locais, de serviço e os módulos de atualização são os únicos componentes do sistema com localidade definida. Os três componentes deverão ficar na máquina que estiver executando os componentes ou serviços sob suas responsabilidades. Um agente local controla agentes de serviço. Estes controlam módulos em geral. No entanto, cada um deles deverá estar

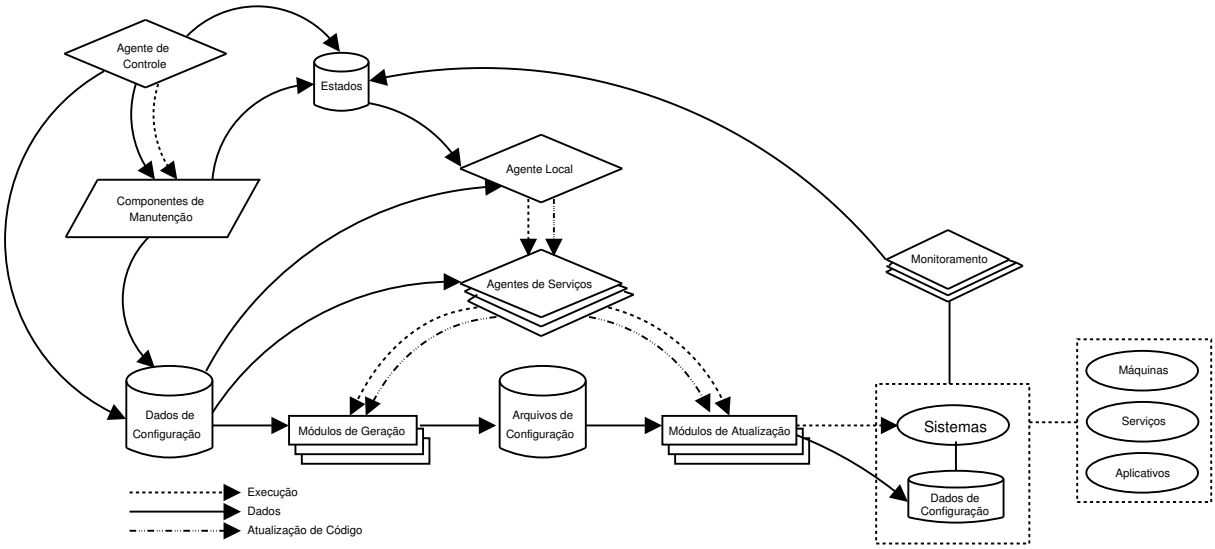

Figura 3.9: Visão do sistema

na máquina que executa os serviços cujos módulos de atualização pertencem ao seu domínio de atuação.

Os componentes do sistema funcionam de forma independente. Um componente não depende de informações de execução ou notificação de um outro componente. Todos os componentes procuram seus dados diretamente na base de dados do sistema. Logo, podem existir apenas dependências de dados na base de dados. Com essa arquitetura é possível distribuir a carga de trabalho do sistema bem como o tráfego de dados entre os componentes e a base de dados, alocando os agentes e módulos (não dependentes de localidade e que exigem mais processamento ou dados) em máquinas distintas.

As próximas seções apresentam a arquitetura dos novos componentes, e agentes de serviços (abordados resumidamente na seção 3.1.3.1), em pormenores.

3.2.1 Agentes

Os agentes são divididos nas seguintes categorias:

Serviços: atualizam e executam módulos do sistema;

Locais: controlam e mantêm o código dos agentes de serviço;

Manutenção: realizam diversas operações de manutenção sobre a base de dados;

Controle: comandam a execução de componentes de manutenção do sistema;

Monitoramento: reportam a atividades de serviços e máquinas da rede.

Os agentes locais têm como finalidade diminuir a complexidade dos agentes de serviço e de monitoramento. Cada máquina conterá apenas um agente local. Este irá consultar na base

de dados os demais agentes atribuídos à máquina, verificando e atualizando o código destes componentes quando necessário, além de controlar a execução periódica dos mesmos.

Um agente de controle é um componente semelhante a um agente local. A diferença é que ele possui níveis maiores de acesso e controle sobre a base de dados. Os componentes de manutenção, por exemplo, são comandados pelo agente de controle. Outro diferencial é que um agente de controle, diferente de um agente local, é livre de localização, podendo ser executado em qualquer máquina da rede.

As seções seguintes abordam detalhes sobre a arquitetura e aplicação dos diferentes tipos de agentes do `ida`.

3.2.2 Agentes Locais

Os agentes locais são responsáveis pelo gerenciamento dos agentes de serviço. Logo, podem ser interpretados como um conjunto de agentes intermediários. Apesar desta aplicação, uma de suas funções é retirar parte da complexidade dos agentes de serivços.

As funcionalidades dos agentes locais são estritamente definidas. Com isso espera-se que operações de manutenção futura nestes componentes não sejam necessárias. No entanto, caso essas atualizações venham a ocorrer, os agentes locais acompanham um mecanismo de re-execução de seu próprio código. Os agentes locais servem ainda como um bom meio de distribuição de agentes e conseqüentes módulos de manutenção de configurações de serviços.

A função de um agente local é verificar o código dos agentes de serviço e monitoramento, carregar o código atualizado para o sistema de arquivos local caso necessário e executar os agentes de serviços sob sua responsabilidade. Cada computador ou equipamento de rede terá um agente local rodando. A partir disso é possível definir dinamicamente os agentes de serviço que o agente local irá controlar/executar.

A figura 3.10 apresenta um fluxograma do funcionamento dos agentes locais. Estes esperam por um sinal de execução ou até um contador de tempo de um dos agentes de serviço sob seu controle chegar a zero. Em ambos os caso, um ou mais agentes de serviço serão executados.

A figura 3.11 apresenta uma ilustração dos agentes locais. Este agente poderá executar, além dos agentes de serviços, agentes de monitoramento, ou seja, o monitoramento de serviços poderá ser distribuído pela rede. Um agente de monitoramento pode ser atribuído a qualquer agente local. Logo, qualquer máquina da rede poderia estar executando um determinado agente de monitoramento.

Para atender os pedidos de execução imediata, os agentes locais, além do funcionamento padrão através dos períodos de tempo dos agentes de serviços, possuem também um canal de comunicação que escuta pedidos de ativação vindos do sistema central (mais especificamente, enviados por uma ferramenta específica). Este seria o caso de uma atualização imediata de serviço, onde o administrador do sistema solicita a ativação imediata de agentes de serviço. A figura 3.12 ilustra graficamente o funcionamento dos agentes locais com suporte a ativação imediata.

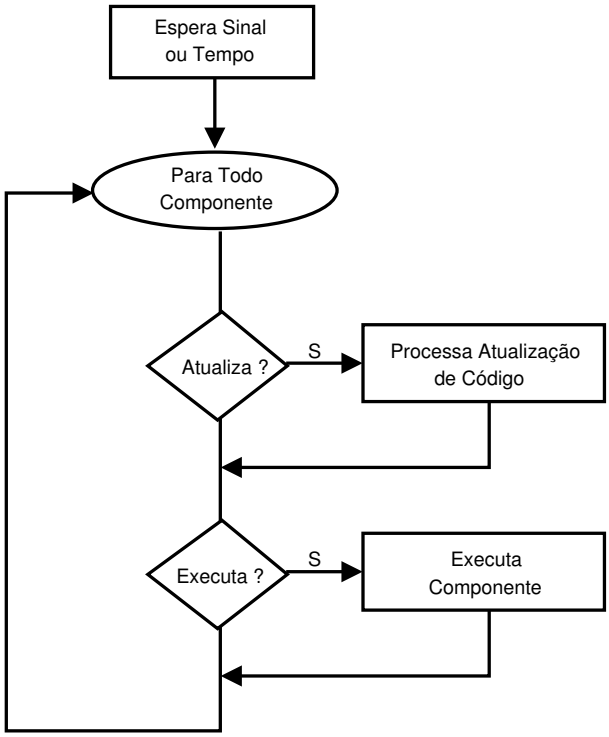

Figura 3.10: Fluxograma de um agente local

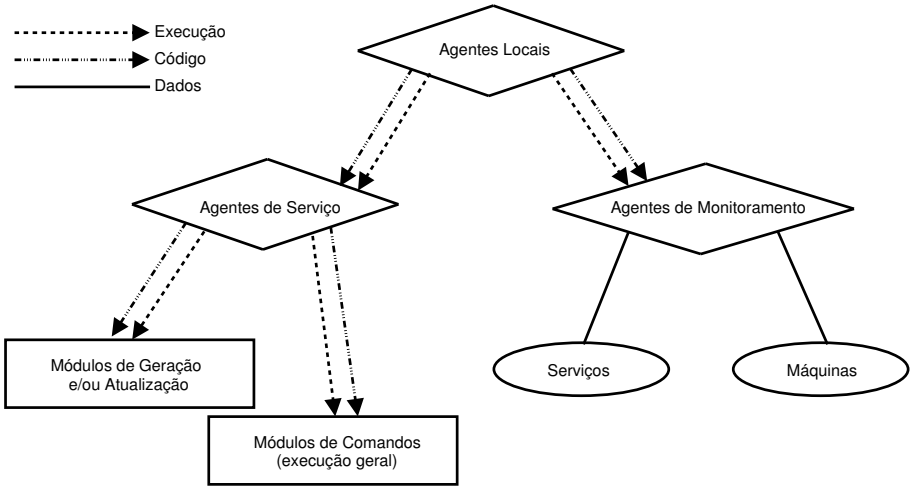

Figura 3.11: Detalhes dos Agentes Locais

Figura 3.12: Ilustração dos agentes locais com ativação imediata

Ao receber o sinal de ativação imediata o agente local inicia as consultas a base de dados e procede a execução dos agentes de serviço caso necessário. A execução é realizada como no caso de um período programado de tempo.

Para simplificar o agente de serviço, o agente local é quem controla as freqüências de tempo e os sinais de execução imediata destes componentes. Além disso, o agente local facilita processos de migração de serviços (ou conjuntos de serviços, onde bastaria migrar apenas um agente de serviço para um novo agente local). Enquanto isto, os agentes de serviço controlam outras dependências como a atualização de dados entre módulos de geração e atualização. Devido a isso, os dois agentes ficam mais simples e fáceis de manter.

3.2.3 Agentes de Serviços

Os agentes de serviços têm como tarefa verificar a necessidade ou não de executar os módulos de geração e atualização, além de transferir o código dos módulos (da base de dados para o sistema de arquivos local) quando necessário. A cada atualização de código dos módulos de geração ou atualização no sistema (base de dados) os agentes de serviço serão alertados sobre as mudanças e carregarão os códigos atualizados para posterior execução. Após uma execução bem sucedida de um módulo de geração o agente de serviço incrementa a chave de atualização (atributo que indica o estado atual de uma entrada de dados, normalmente identificado como um número que aumenta a cada atualização de dados sobre um registro de informações da base de dados) dos arquivos de configuração do respectivo serviço. Com isso, o agente de serviço responsável pelo módulo de atualização do serviço em questão irá detectar a mudança na chave de atualização e conseqüentemente executará o módulo de atualização correspondente.

A figura 3.13 apresenta os agentes de serviços em maiores detalhes. Como pode ser observado, entre os agentes de serviços e a base de dados existe tráfego de dados e comandos. Os agentes de serviço são responsáveis por atualizar o código do módulos e indicar a atualização bem sucedida a base de dados, evitando uma nova atualização na próxima execução.

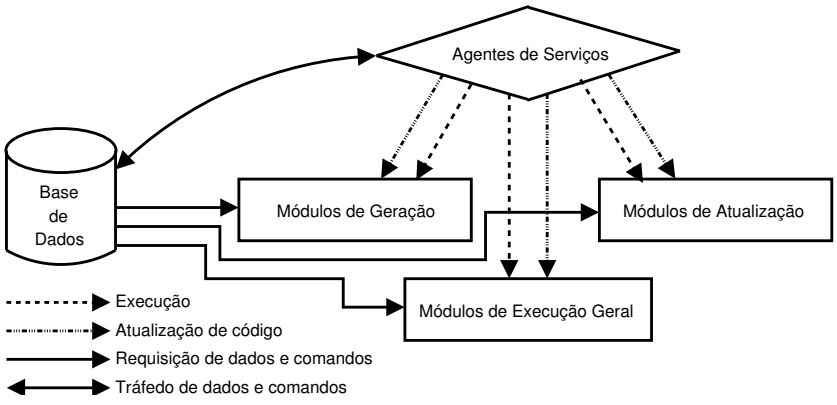

Figura 3.13: Agentes de Serviços em Detalhes

Os agentes de serviços executam os módulos sob sua jurisdição. É na base de dados que estão indicados quais são estes componentes.

O fluxograma de um agente de serviço é semelhante ao de um agente local (ver figura 3.10). Com excessão que o primeiro é controlado pelo segundo e, por isso, não precisa esperar por um determinado período de tempo ou por um sinal para execução imediata.

3.2.4 Agentes de Controle

Os agentes de controle possuem como funções básicas executar e controlar componentes de manutenção do sistema propriamente dito. Para tanto, possuem um maior nível de acesso ao dados e recursos do sistema.

Um agente de controle controla a execução de componentes de manutenção. Estes componentes são responsáveis por tarefas importantes no sistema. Entre elas poderia ser citado o controle da consistência e integridade dos dados da base de dados e o gerenciamento e ativação das atualizações, carregadas pelos agentes de serviços. Por isso, esses agentes são considerados componentes críticos, seguros e confiáveis. Logo, devem executar em ambientes adequados e restritos a um domínio administrativo seguro.

A figura 3.14 apresenta uma visão geral dos agentes de controle. Entre os componentes sob o comando dos agentes de controle estão os módulos e agentes de manutenção geral do sistema. A abrangência vai desde simples atualizadores de código da base de dados até verificadores e ajustadores de dependências de dados.

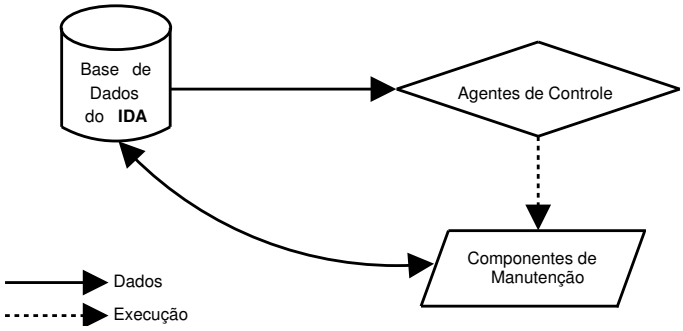

Figura 3.14: Agentes de Controle

3.2.5 Componentes de Manutenção

Os componentes de manutenção do sistema realizam tarefas essenciais como a verificação e validação de dependências de dados entre as entradas da base de dados, manutenção de contadores globais, análise de dados de monitoramento, análise do estado dos dados de configuração e a catalogação de referências a dados em entradas de serviços.

3.3 Exemplo de Aplicação Sobre uma Rede

O objetivo desta seção é apresentar um exemplo prático de aplicação dos componentes vistos neste capítulo, sobre um determinado conjunto de computadores. A figura 3.15 ilustra a aplicação de diversos componentes do `ida`.

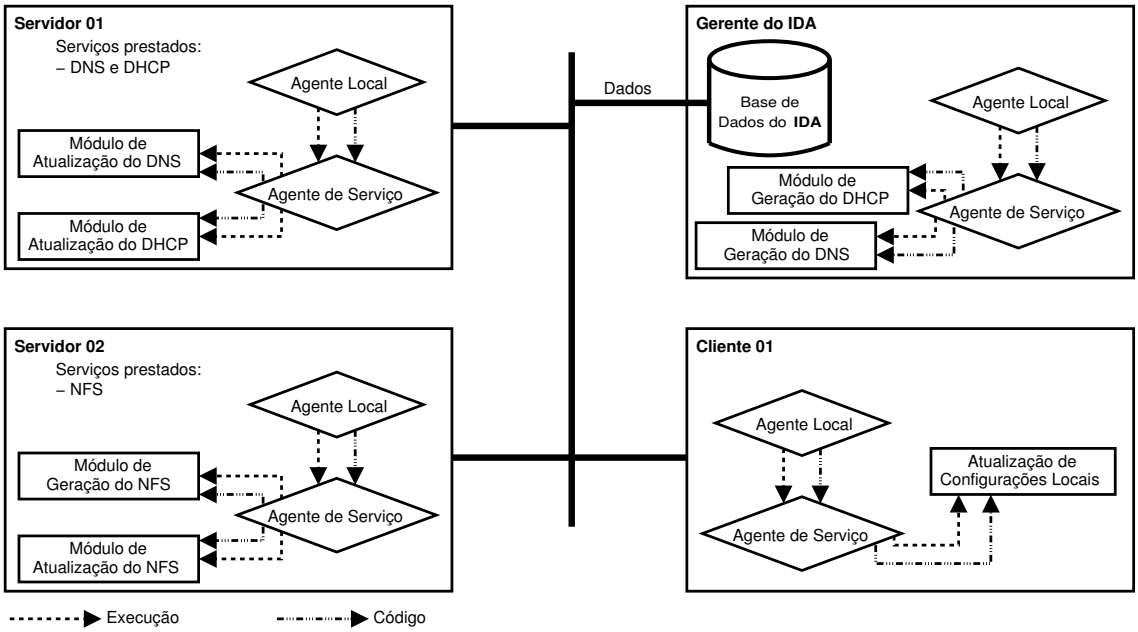

Figura 3.15: Visualização de uma aplicação prática dos componentes do `ida`

O ambiente mostrado na figura 3.15 é composto por dois servidores, uma máquina cliente e uma máquina administrativa. Em cada computador há um agente local e um agente de serviço.

O servidor 01 é responsável pelos serviços de DHCP e DNS. Logo, os módulos de atualização destes serviços estão sob o comando do agente de serviços atribuído ao agente local desta máquina. Os módulos de geração dos arquivos de configuração do DNS e DHCP estão sendo executados no computador administrativo. Este seria um exemplo de distribuição de componentes e conseqüente distribuição (ou concentração) de processamento e tráfego de dados pela rede. No caso do servidor 02 ambos os módulos de geração e atualização do NFS são executados localmente. Na máquina cliente 01 é executado apenas um módulo de manutenção das configurações locais da máquina.

Uma alteração do registro do número IP da máquina cliente na base de dados causaria a execução do módulo de atualização das configurações locais da máquina. A alteração nos dados é percebida pelo agente de serviço e este imediatamente ativa o módulo de atualização correspondente. No caso, a entrada da máquina na base de dados contém um atributo que indica a alteração ou não de dados. Este atributo é automaticamente atualizado pelo editor de entradas de dados (da base de dados) do sistema.

No ambiente apresentado todos os componentes são funcionalmente independentes e se conectam diretamente com a base de dados para obter informações necessários sobre configurações e obrigações atribuídas a cada um. Apesar de os agentes de serviços serem executados pelos agentes locais (em uma configuração normal), eles poderiam ser controlados através de agendadores de tarefas locais a uma máquina. Porém, neste caso funcionalidade como ativação imediata e atualização do código dos agentes de serviços seriam perdidas. Mesmo assim, esta é uma opção disponível e que cabe a cada administrador do sistema optar por ela ou não, caso

achar conveniente.

3.4 Base de Dados

A base de dados é um ponto importante do sistema. É nela que são mantidas todas as informações necessárias para descrever, configurar e manter serviços, máquinas e sistemas em geral.

Para permitir um bom nível de integração, relação e dependência entre os dados é aconselhável o uso de uma base de dados unificada. Essa é a abordagem utilizada por sistemas como o NetInfo (seção 2.3.3), que permite a integração do gerenciamento de praticamente todos os aplicativos e sistemas do ambiente Mac OS X. O `ida` utiliza uma abordagem semelhante ao do NetInfo para a base de dados.

A configuração e arquitetura da base de dados é adaptável ao contexto administrativo ou as necessidades de diferentes domínios administrativos. Com isso, o administrador do sistema têm a possibilidade de especificar a organização e distribuição dos dados, garantindo que as particularidades de seu domínio administrativo sejam contempladas.

A seguir são descritos os dados armazenados, características de configuração e recursos de acesso à base de dados. O apêndice B apresenta algumas informações a mais sobre os dados da base de dados. Nele podem ser encontrados exemplos de registrados cadastrados na base de dados do `ida`.

3.4.1 Descrição dos Dados

A figura 3.16 apresenta uma visualização da arquitetura interna da base de dados do `ida`. Como pode ser observado, ela é organizada em várias sub-árvores de dados.

Cada ramo da árvore de dados contém informações de determinado tipo. Na seqüência são descritas as características e os principais tipos de dados armazenados em cada sub-árvore da base de dados. Maiores detalhes e exemplos práticos podem ser vistos nos apêndices B e D.

Dados: contém dados de descrição e configuração de máquinas e serviços;

- **Máquinas:** dados referentes à descrição de computadores de uma rede. Os dados incluem endereço IP, nome, aliases, configurações gerais de rede (sub-rede, máscara, *gateway*, DNS), grupos a qual pertence, descrição geral das características da máquina (informações sobre *hardware*), localização, identificação de patrimônio (especialmente útil para instituições públicas), sistemas operacionais, serviços prestados a rede e informações gerais. As descrições das máquinas são dados básicos para a manutenção das configurações de serviços de rede como o DNS, DHCP e IPTables e as configurações locais dos próprios computadores.
- **Serviços:** esta sub-árvore contém as informações necessárias para a configuração e manutenção de serviços de uma rede de computadores. Estas informações, juntamente

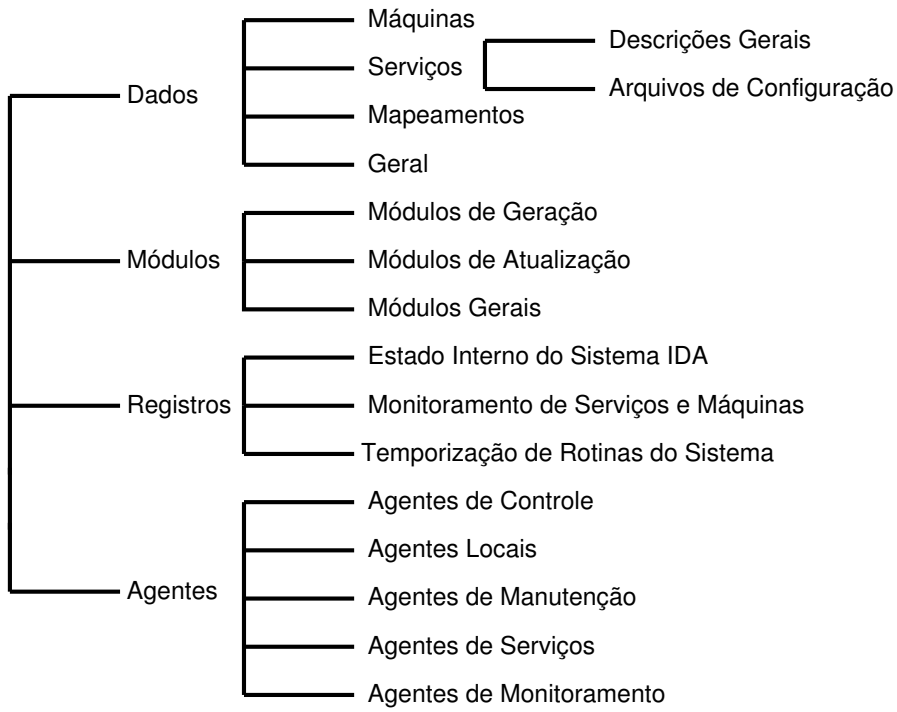

Figura 3.16: Visualização da base de dados

com as descrições de máquinas, servem como base para os módulos de geração. Estes irão produzir os arquivos e descrições de configuração para serviços, máquinas e sistemas no geral. Os dados resultantes serão utilizados pelos módulos de atualização.

Mapeamentos: informam qual (quais) máquina executa cada serviço. Os mapeamentos servem para abstrair relações e dependências entre os dados utilizados para a configuração de um serviço e a sua localização (máquina hospedeira). Além disso, facilitam a manutenção de atualizações.

Geral: descrições gerais de configuração, representação e hierarquia são mantidas nesse ramo da base de dados. No geral, estes dados são utilizados e estão acessíveis a todos os componentes do sistema. Logo, informações importantes e confidenciais não devem ser mantidas nessa sub-árvore.

Módulos: a sub-árvore de módulos contém informações de funcionamento e o código de execução dos módulos do sistema. Estes são classificados basicamente em três categorias: configuração e geração, atualização e gerais.

Agentes: a sub-árvore de agentes armazena dados, configurações, características e código fonte dos agentes do sistema.

Controle: agentes com alto nível de confiabilidade e autorização de acesso a base de dados. As entradas descrevem os componentes sob o comando dos agentes de controle e dados de execução como intervalo de execução e diretório para o armazenamento

do código de execução de componentes que ficarão dentro do seu domínio de atuação;

Locais: as entradas destes agentes contém dados de identificação, localização e agentes de serviço sobre o seu comando. Estes agentes facilitam a manutenção, inclusão e migração de agentes de serviços de um local ao outro;

Manutenção: estas entradas indicam aos seus respectivos componentes seus intervalos de execução, as árvores de dados sobre as quais deverão atuar e mantém informações referentes ao código fonte destes agentes;

Serviços: as entradas da base de dados indicam quais módulos estarão sob o controle destes agentes, o diretório local para o armazenamento e posterior execução dos módulos e o intervalo de freqüência de execução, além do código fonte do agente;

Monitoramento: contém informações necessárias para a execução de um determinado agente de monitoramento sobre um determinado serviço. Entre estes dados podem estar as identificações dos módulos de geração e atualização responsáveis pelo serviço em específico e o tempo de execução.

Registros: Os registros armazenam estados do sistema, estados de máquinas e serviços e estatísticas de tempo de execução de rotinas do sistema;

Estado Interno do Sistema: os dados de estado do sistema reportam informações de execução de todos os componentes distribuídos do sistema. Com isso é possível identificar falhas e problemas de execução em geral e extrair um relatório geral de atividades dos módulos e agentes do sistema.

Monitoramento de Serviços e Máquinas: Os registros de monitoramento servem para acompanhar o estado de serviços, máquinas e sistemas em geral. Diagnósticos, detecção e correção de falhas na prestação de serviços são alguns dos objetivos derivados dos registros de monitoramento.

Temporização de Rotinas do Sistema: tempo gasto por rotinas internas do sistema. Com esses dados também é possível detectar a sobrecarga de uma determinada máquina, visto que ela pode estar levando muito tempo (levando em consideração a média geral ou casos específicos) para executar rotinas simples;

Alguns dos objetivos inerentes às ramificações de dados são a organização e distribuição das informações, controle granular (individual, restritivo e dinamicamente configurável) de acesso aos dados, segurança e hierarquia. Os componentes do `ida` terão acesso restrito e controlado sobre os dados do sistema. Um módulo de atualização, por exemplo, terá acesso apenas as entradas de dados que contém os arquivos de configuração do serviço mantido pelo módulo.

3.4.2 Interface Única

O `ida` tem uma única interface de comunicação com a base de dados. Todos os componentes do sistema utilizam essa mesma interface. Desta forma, basta reimplementar a interface, no caso de o administrador da rede desejar utilizar uma outra base de dados, e todos os componentes continuarão a funcionar sem maiores problemas.

Outra funcionalidade da interface única é a definição de novos servidores para a base de dados. Neste caso, basta substituir na implementação da interface a variável que indica quais são os novos servidores de dados que os componentes do sistema passarão automaticamente a utilizar os novos servidores de dados.

3.4.3 Vantagens da Unificação dos Dados

Uma base de dados única traz algumas vantagens em relação ao uso de múltiplos arquivos de configuração ou descrição de configuração. Manter arquivos distintos, principalmente se forem em formatos distintos (o que é comum em ambientes de gerenciamento e mesmo em muitas das ferramentas disponíveis) pode levar a inconsistências e erros de configuração de serviços e máquinas. Quanto maior e mais diverso for o domínio administrativo, maior é a probabilidade de falhas e divergências de configuração ocorrerem. Por outro lado, o uso de uma única base de informações, com relação e dependência de dados, facilita o gerenciamento de sistemas e diminui consideravelmente as possibilidades de erros. Em um ambiente desse gênero as descrições de configuração de serviços e máquinas podem ser geradas por ferramentas automáticas e menos propícias a falhas. Além disso, uma base de dados unificada permite, com maior facilidade, a construção de instrumentos automáticos de verificação de consistência e integridade dos dados armazenados, sendo um recurso a mais para evitar possíveis falhas humanas.

A integração do gerenciamento da rede é mais simples com o uso de uma base de dados única. Além disso, esta pode ser centralizada ou distribuída, proporcionando escalabilidade e disponibilidade. Os diversos componentes do sistema necessitam apenas ter acesso aos dados de configuração e gerenciamento.

A própria sobreposição de dados, apresentada rapidamente na próxima seção, é um exemplo de outra vantagem de uma base de dados única.

3.4.4 Mapeamentos e Sobreposição de Configurações

Uma das características importantes e úteis do sistema é a sobreposição de informações. Ela consiste em ter dados que servem de referência geral, permitindo a especialização de configurações individuais a cada máquina (ou a cada programa servidor) que executa o mesmo serviço. Esse recurso pode também ser utilizado apenas para a configuração de uma única instância de um serviço. Neste caso, a descrição geral poderia manter os dados de configuração padrão e somente as configurações específicas são definidas no registro (na base de dados) do serviço. Isso facilita e agiliza a especificação de configurações gerais e particulares de serviços e sistemas em

geral. A seção 4.4.2 apresenta alguns exemplos de uso da sobreposição de configurações.

Outra característica igualmente importante é a independência entre quem usa a informação (serviços e máquinas) e quem processa a informação (busca na base de dados e formata de acordo com o serviço). Neste contexto, os mapeamentos fornecem um meio de descrever de forma abstrata e independente as relações entre máquinas e respectivos serviços prestados. Um serviço, como o DHCP, é atribuído a uma máquina. Esta relação (quem é a máquina que executa o DHCP) é descrita em um mapeamento. Desta forma, para trocar de lugar o serviço bastará alterar o mapeamento indicando a nova localização (servidor) que os demais sistemas serão automaticamente ajustados. Este poderia ser um caso de migração do serviço de DHCP para uma nova máquina. A alteração da entrada de mapeamento implicará na atualização automática de todas as configurações que mantenham alguma relação com o servidor DHCP. Esse processo é controlado e ativado, de forma automática, pelos agentes de manutenção do sistema.

3.5 Características Incorporadas ao `ida`

Na seção 2.3.6 foram apresentadas diversas características desejáveis em um sistema integrado de gerenciamento de equipamentos e serviços de rede. Nesta seção, é apresentado como o `ida` incorpora algumas dessas características. Com isso, espera-se construir um sistema escalável, flexível e adaptável aos diversos contextos de diferentes domínios administrativos. Dentro deste contexto, as próximas seções apresentam brevemente algumas das principais características do sistema.

3.5.1 Escalabilidade

O sistema tem por objetivo ser escalável. No contexto do `ida`, isso significa que é possível utilizar o sistema para gerenciar praticamente qualquer número de serviços ou dispositivos de redes de pequeno, médio e grande porte [3].

A escalabilidade do `ida` é inerente ao núcleo do sistema (agentes de controle e manutenção, base de dados, módulos de geração e árvore de diretórios dos códigos fontes dos componentes do sistema). Como cada componente é independente dos demais ele pode ser executado em qualquer máquina da rede, possibilitando distribuição da carga de processamento e organização da segurança e confiabilidade do sistema. Todos os componentes principais do sistema podem estar sendo executados em máquinas dedicadas (restritas e bastante seguras), ou em uma única máquina.

Devido a essas características de escalabilidade (disponibilidade e distribuição) o sistema pode ser adaptado a redes de praticamente qualquer tamanho, com praticamente qualquer número de equipamentos. O `ida` pode ser implantado de acordo com as necessidades e exigências de cada administrador de domínio.

[3] Considerando que **redes de pequeno** porte atíngem apenas algumas dezenas de equipamento; **redes de médio** porte algumas centenas de equipamentos e **redes de grande porte** alguns milhares de equipamento.

Em uma rede de grande porte, é possível estabelecer domínios de acesso à base de dados e assim distribuir tarefas administrativas. Cada sub unidade administrativa pode ficar responsável pelos componentes que atuam nos seus domínios. Com isso tem-se um gerenciamento ao mesmo tempo centralizado e distribuído. Além disso, o servidor de dados pode ser distribuído ou replicado, visando garantir escalabilidade e disponibilidade de dados.

3.5.2 Flexibilidade, Praticidade e Independência

A concepção do `ida` reside sobre a prerrogativa de que todo administrador de redes tem suas práticas e preferências administrativas. Com isso em mente, a proposta do sistema é uma ferramenta bastante flexível, para que possa ser facilmente adaptada aos mais variados tipos de ambientes, políticas administrativas, sistemas e administradores de rede.

Entre as características que levam adaptabilidade ao sistema estão a flexibilidade, praticidade e independência. Praticamente todos os componentes do sistema são de fácil manutenção e atualização. Isso por que todos são logicamente independentes. Cada componente do sistema busca informações apenas na base de dados. Inclusive os códigos de execução de agentes e módulos são mantidos na base de dados, ou seja, até a instalação, configuração e ativação do sistema são praticas simples e rápidas.

O sistema possui uma razoável organização e divisão de tarefas e funcionalidades devido a hierarquia de agentes, módulos e independência entre eles. Isso torna o `ida` uma ferramenta relativamente fácil de ser mantida.

O administrador do domínio, para incluir o gerenciamento de um novo serviço, precisará definir os componentes básicos para a manutenção das configurações e atualização do serviço. Isso é feito através de módulos de geração e atualização (ver seção 3.1.2), que podem ser desenvolvidos em diferentes linguagens de programação (ver seção 4.2).

3.5.3 Consistência e Integridade dos Dados

O sistema dispõe de um mecanismo (opcional) de sincronização. Sua utilização pode ser feita automaticamente pela interface de comunicação entre o sistema e a base de dados ou através da invocação, por parte do usuário ou desenvolvedor, dos métodos de sincronização da interface (ver maiores detalhes sobre a interface na seção D.1 do apêndice D). O mecanismo de sincronização inibirá a leitura dos dados atualmente bloqueados para leitura/escrita por algum componente.

3.5.4 Adaptação e Extensibilidade

O sistema é independente de plataforma e linguagem de programação. O `ida` poderá ser utilizado sobre "qualquer" sistema operacional ou equipamento que tenha suporte as linguagens de programação Java, Perl, Python, C, C++, e *Shell Script* (e outras mais, descritas no apêndice C, seção C.1). Além disso, será adaptável a praticamente "qualquer" ambiente e "qualquer" conjunto de agentes e/ou módulos. Com a interface padrão de comunicação entre o sistema e

a base de dados é fácil adaptar o sistema para utilizar novas bases de dados ou incluir módulos com funcionalidades adicionais.

3.5.5 Tolerância a Falhas

Podem existir n réplicas da base de dados do `ida`. Quando o servidor primário sai de funcionamento o secundário automaticamente assume a posição e o sistema continua funcionando normalmente.

A interface de conexão entre os componentes e a base de dados do sistema possui uma lista de servidores declarada. Esta lista deve ser composta pelos servidores (da base principal do sistema) primário, secundário e assim por diante.

O módulo de controle de replicação e tolerância a falhas monitora o servidor primário da base de dados do `ida`. No caso de uma falha visível e crítica o servidor auxiliar torna-se o primário. Caso a conexão falhar com o servidor primário, a própria interface de conexão irá tentar se conectar com os demais servidores.

Além da tolerância a falhas do próprio sistema um controle de falhas referentes a serviços gerenciados pelo `ida`. O objetivo é fornecer recursos para manter a disponibilidade mesmo em condições adversas.

Disponibilidade de serviços. Quando detectada uma falha em um serviço e houver definida uma política de recuperação e/ou migração o sistema irá ativar o serviço no servidor auxiliar (secundário) e adaptar todos os demais sistemas para que eles passem a utilizar a nova localização do serviço como primário.

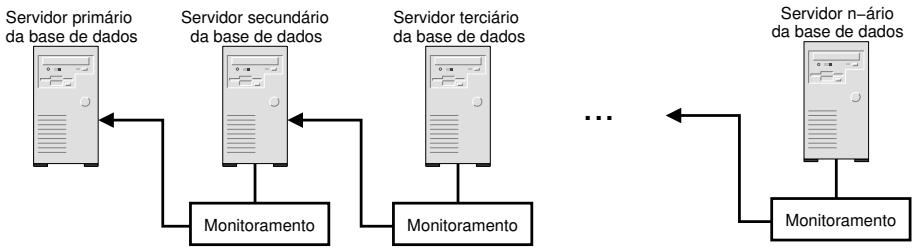

Figura 3.17: Possível configuração de tolerância a falhas da base de dados do `ida`

Assim que for detectada a falha do servidor primário o secundário irá assumir sua posição. Serviços como DNS serão imediatamente atualizados. Com isso a conexão ao servidor primário (antigo servidor secundário) continuará normal.

3.5.6 Segurança

O núcleo do sistema é praticamente o único ponto de falha crítico do `ida`. Logo, fica fácil monitorar e manter a segurança do sistema. Além disso, a violação ou adulteração de um componente não afeta diretamente o funcionamento do sistema, ou dos demais componentes. Isso por que a maioria dos componentes têm acesso restrito e individual a dados e configurações do sistema.

A comunicação entre os diversos componentes e a base de dados pode ser rigorosamente controlada. Cada componente poderá ter seu acesso limitado exclusivamente as áreas de dados bastante específicas. Com isso, informações mais importantes do sistema somente poderão ser vistas por quem realmente as necessita. Essa rigorosidade é opcional e definida pelo administrador do sistema.

A conexão entre componentes e base de dados pode ser feita através do uso de TLS ou SSL. Essa é outra opção definível pelo administrador do sistema. Para isso, basta configurar a interface de comunicação do `ida`.

O controle de acesso e autenticação é realizado através do protocolo de comunicação e regras de acesso à base de dados. A autenticação é realizada através de senhas de acesso, que são uma espécie de chave individual a cada componente do sistema. Cada componente pode acessar (ler ou modificar) dados da base de dados conforme as regras de acesso, que definem quem pode fazer o que e sobre o que. Exemplo: um determinado componente tem acesso de leitura apenas a uma sub-árvore de dados e pode modificar apenas alguns atributos específicos das entradas de dados desta sub-árvore.

3.5.7 Granularidade Administrativa

Um aspecto importante de qualquer sistema de gerenciamento de redes são as políticas gerais de segurança, onde é necessário uma detalhada interpretação e análise por parte dos administradores de sistemas (MOFFETT; SLOMAN, 1993). As políticas de acesso vão desde acessos administrativos (por usuários e gerentes do sistema) até o acesso aos dados por parte dos mais diversos componentes do sistema.

O `ida` tem políticas de controle de acesso que proporcionam definições refinadas e complexas de restrições e permissões de acesso a base de dados. Podem ser estabelecidos vários níveis de acesso. Um usuário, por exemplo, pode ter acesso a toda base de dados, a um ramo em específico, a várias sub-árvores, a entradas específicas ou apenas a alguns atributos (não importando a entrada). Com esses recursos é possível projetar níveis de acesso limitados às necessidades de cada administrador do sistema.

Além das possibilidades de controle de acesso anteriormente descritas é possível estabelecer-se o tipo de ação possível. Usuários e componentes do sistema podem ter acesso apenas de leitura a determinados dados. Acessos de escrita são garantidos apenas a usuários e componentes considerados seguros e confiáveis. Acessos apenas de consulta (sem possibilidade de leitura ou escrita) podem ser úteis no caso de componentes que precisam identificar existências ou quantidades. Esse seria o caso da verificação de novos serviços ou do número de serviços e máquinas constantes na base de dados.

3.5.8 Automatização

O `ida` tem um funcionamento dinâmico e automático. Agentes podem ser programados para executar automaticamente. Quase todos os componentes do sistema podem ser carregados

e ativados dinamicamente. As execuções são controladas pelos agentes de controle, locais e de serviços. No caso de se querer acrescentar um novo módulo, por exemplo, basta incluí-lo na base de dados e alterar a entrada do agente correspondente. O novo componente terá seu código atualizado (pelo agente responsável) e será automaticamente executado. Logo, os serviços e máquinas podem ser mantidos de forma simples, dinâmica e automática. Realizar alterações na base de dados é o suficiente para um componente do `ida` ser atualizado ou incluído no sistema. O mesmo é válido para o gerenciamento de serviços, máquinas e sistemas em geral. Basta registrar novos módulos na base de dados, referenciá-los a algum agente de serviço que automaticamente estarão ativos no sistema.

3.5.9 Políticas de Gerenciamento

> Política é uma especificação persistente de um objetivo a ser atingido ou um conjunto de ações a serem executadas em um tempo futuro ou como uma atividade regular em andamento (MOFFETT; SLOMAN, 1993).

Algumas modalidades de políticas incluem (MOFFETT; SLOMAN; TWIDLE, 1990):

políticas imperativas: fornecem ao agente a imperativa de carregar uma ação se este tiver o poder para tanto. Em grande parte dos casos uma política imperativa é uma obrigação.

políticas autoritativas: provêem ao agente o poder para executar uma ação, isto é, as operações permitidas sobre um objeto alvo. Regras de acesso são exemplos de políticas de autorização (MOFFETT; SLOMAN; TWIDLE, 1990).

O `ida` utiliza ambos os tipos de políticas. Agentes de manutenção, por exemplo, têm a obrigação de anunciar aos devidos agentes de serviços a necessidade de executar um determinado módulo de geração a fim de proceder a atualização de uma máquina ou serviço. Para tanto, o agente de manutenção precisará também ter acesso aos dados que identificam esta necessidade de atualização.

A exemplo, políticas autoritativas são aplicadas sobre agentes locais e de serviços. Estas políticas definem o nível de acesso e operação destes agentes sobre um determinado componente do sistema (agente ou módulo).

Outras políticas possíveis no sistema são as políticas de manutenção e disponibilidade de serviços. Neste caso, o administrador do sistema tem o poder de definir quais ações serão tomadas por agentes de controle no instante em que falhas de prestação de serviço forem detectadas. Isso pode incluir a atualização do serviço, a re-inicialização de serviços ou a migração do serviço. Cada uma desta ações pode ser programada para ser executada várias vezes consecutivas. Por exemplo, várias tentativas de atualização de serviço antes de uma migração.

3.5.10 Organização e Planejamento

O ida leva os administradores de redes a planejarem e organizarem a estrutura e gerenciamento de suas redes. Como o sistema preza pela flexibilidade, adaptação e liberdade, alguns aspectos de organização e planejamento são necessários para a estruturação e implantação do ida.

O primeiro passo é identificar o que se pretende gerenciar na rede. Com essa etapa concluída, parte-se para a fase de definição e descrição dos dados que serão necessários e armazenados na base de dados. Esta fase é bastante importante e implicará no funcionamento posterior do sistema. Logo, um bom planejamento e organização são fundamentais a fim de evitar problemas futuros.

Previsão de escalabilidade e adaptação de novos serviços e dispositivos de rede também devem ser pensados e levados em consideração na fase de estruturação da base de dados. Por isso, é importante que o administrador da rede e/ou responsável pela implantação do sistema tenha um bom e detalhado conhecimento da rede.

3.6 Resumo de Algumas Vantagens do ida

A seguir são resumidas algumas das vantagens do ida:

Base de dados dinâmica: permite a inclusão, remoção ou alteração de dados em tempo de execução, possibilitando um controle dinâmico dos componentes do sistema e permitindo a definição de novos tipos de dados;

Auto atualização: além de atualização automatizada de máquinas e serviços o ida é acompanhado de mecanismos que controlam a atualização dos próprios componentes do sistema. Esses recursos facilitam a manutenção e extensão do sistema propriamente dito;

Auto diagnóstico: os componentes do ida são programados para reportar seu estado em tempo de execução. Falhas de acesso, erros de escrita, permissões inapropriadas, execução bem sucedida, falhas de atualização, erros nas chamadas de sub-rotinas e identificadores de processo são alguns dos dados reportados a base de dados pelos componentes do sistema. Com esses dados é possível identificar e diagnosticar agentes e módulos problemáticos. A localização (máquina hospedeira) do componente é também automaticamente registrada nos relatórios de execução.

Independência de plataforma e linguagem de programação: linguagens de programação como Java, Perl, Python e *Shell Scripts* são utilizadas no sistema. Estas linguagens são interpretadas ou rodam sobre uma plataforma de execução, provendo um bom grau de independência de plataforma. Além disso, a extensão do sistema pode facilmente ser realizada utilizando uma dentre as várias linguagens comportadas pelo sistema (ver seção 4.2). O limite é a linguagem de programação utilizada.

O `ida` fornece mecanismos e facilidades que outras ferramentas não disponibilizam, como auto monitoramento, atualização e temporização, tolerância a falhas em vários níveis, granularidade administrativa, liberdade de plataforma e linguagem de programação e variedade de ferramentas de apoio (no apêndice C são apresentados alguns sistemas auxiliares que podem ser utilizados com o `ida`). Este sistema fornece também um meio de integração de gerenciamento entre diferentes ferramentas de administração ou configuração em geral.

Em uma única base de dados é possível manter todas as informações e dados da rede, tanto para simples relatórios e controle de equipamentos e serviços como para manutenção e configuração de sistemas. A unificação das informações da rede permite um gerenciamento completo sobre os mais diversos dispositivos e serviços da rede.

3.7 Resumo

O presente trabalho têm como foco principal a gerência de configuração, onde dispositivos como servidores, *firewalls* e clientes de um modo geral são configurados e mantidos. Outro aspecto importante é o fato de a arquitetura do sistema proposto ser aberta a integração das demais divisões de gerenciamento.

O `ida` utiliza tanto agentes passivos quanto agentes ativos (que contém um grau de autonomia). Os agentes ativos serão responsáveis pelo controle da manutenção de serviços e dispositivos. Enquanto que os agentes passivos serão responsáveis por tarefas corriqueiras de manutenção no sistema propriamente dito.

A arquitetura proposta não é destinada a substituir ferramentas existentes. Ela foi projetada para suprir algumas deficiências de ferramentas do gênero (que podem ser vistas na seção 2.3, do apêndice A) e com o intuito de integrar sistemas na tarefa de gerenciamento geral de redes de computadores.

O sistema aqui proposto vem a suprir várias da deficiências apontadas na seção 2.4 do capítulo 2. Entre as características abordadas pela proposta do trabalho estão:

- distribuição de componentes e dados;
- granularidade administrativa e de acesso;
- independência de plataforma e linguagem;
- flexibilidade/liberdade/adaptabilidade;
- escalabilidade;
- administração simultaneamente centralizada e distribuída;
- tolerância a falhas (de serviços e do próprio sistema);
- dinamicidade;

- auto instalação/atualização/monitoramento e diagnóstico;

O `ida` é flexível quanto a sua própria configuração e estruturação, podendo o administrador do domínio modificá-la quando necessário ou desejado.

O apêndice B apresenta um exemplo de relação de dados e exemplos de entradas cadastradas na base de dados. Os exemplos incluem a configuração de uma máquina e de um serviço da rede.

4 IMPLEMENTAÇÃO DO SISTEMA

O capítulo 3 apresentou a arquitetura geral do e as principais características incorporadas ao `ida`. Este capítulo apresenta a arquitetura implementada, as tecnologias utilizadas no desenvolvimento do sistema e a interface padrão de conexão entre os componentes do sistema e a base de dados.

Neste capítulo é apresentado apenas um resumo da implementação do sistema. O apêndice D fornece maiores detalhes sobre os recursos implementados no `ida`.

4.1 Arquitetura Implementada do Sistema

A figura 4.1 ilustra a arquitetura base do sistema implementado. Nela podem ser vistos componentes como: agentes de controle, manutenção, monitoramento, local e serviços; módulos de manutenção, geração e atualização.

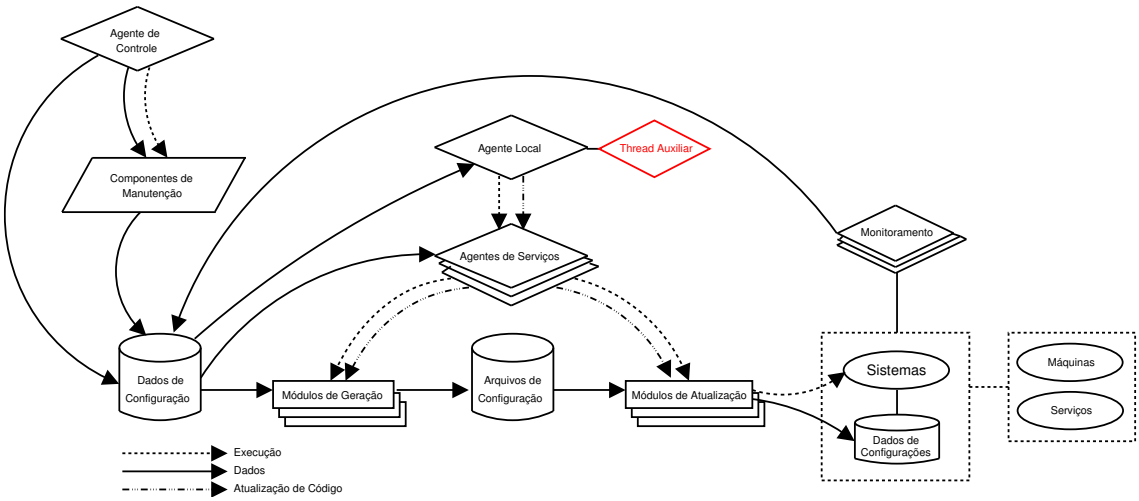

Figura 4.1: Arquitetura Base do Sistema Implementado

Os agentes de controle gerenciam os componentes de manutenção (responsáveis por tarefas administrativas básicas sobre os dados da base de informação do `ida`) e os agentes de monitoramento e migração. Estes monitoram os dados coletados pelos demais agentes de monitoramento, analisam e averiguam a necessidade de medidas de atualização ou migração de serviços baseando-se em políticas armazenadas na base de dados.

Atualmente estão disponíveis módulos para a geração do arquivos de configuração de diferentes serviços. Estes módulos geram os arquivos de configuração e os armazenam na base de dados para posterior uso pelos respectivos módulos de atualização, que residem nas máquinas responsáveis pelos serviços. Cada um desses serviços também possui um módulo de atualização implementado. Este é responsável pela transferência (do núcleo do sistema para a máquina local) e alocação (nos diretórios apropriados) dos arquivos de configuração dos respectivos serviços.

Um tipo geral de agentes local e outro de agente de serviços estão disponíveis para aplicação e uso. Cada agente local, residindo em uma máquina específica, terá uma entrada própria descrevendo os agentes serviços e/ou módulos sob sua responsabilidade. Os agentes de serviços possuem entradas com os módulos sob sua responsabilidade.

Os agentes locais controlam a execução periódica dos agentes de serviços. Eles também possuem um fluxo de execução auxiliar que é destinado a receber notificações de atualização imediata. Estas são mensagens textuais simples que trafegam através de um canal específico de comunicação. Assim que uma mensagem desse gênero por recebida pelo fluxo auxiliar os agentes de serviço serão ativados para verificarem a necessidade ou não de atualização ou execução dos módulos por eles gerenciados.

Os agentes de monitoramento podem ser executados independentemente ou através de agentes locais ou de serviços. Implementações funcionais incluem agentes de monitoramento para os serviços de DNS, DHCP, SSH, MySQL e Apache. Eles analisam e reportam dados de estado à base de dados.

No caso da arquitetura apresentada na figura 4.1 foram utilizados agentes locais para simplificar os agentes de serviços propriamente ditos e facilitar a atualização e controle do código dos agentes de serviço. Como os agentes locais são simples e tem funcionalidades mínimas, a idéia é que estes nunca precisariam ser atualizados. Eles verificam atualizações de código e executarem os agentes de serviços, verificando também o seu próprio código. Caso uma atualização de seu próprio código precise ser feito, o novo código é copiado da base de dados para o sistema local e o agente local se re-executa, garantindo a propagação da atualização de código.

4.2 Tecnologias Utilizadas

A base do sistema foi desenvolvida em Perl. A escolha dessa linguagem deve-se a vários fatores. Entre eles estão a praticidade e agilidade no desenvolvimento de sistemas, aceitação da linguagem, independência de plataforma e desempenho (MARTIN, 2001; OUSTERHOUT, 1998; JAVA, 2000; RICHICHI, 2001) (mais detalhes na seção C.1.1).

Módulos do `ida` podem ser escritos em qualquer linguagem de programação que suporte a interface padrão de comunicação do sistema ou o protocolo LDAP (Lightweight Directory Access Protocol). Este foi o protocolo de comunicação escolhido para o desenvolvimento do sistema.

Apesar de a base do sistema ser em Perl, ela pode ser portada para qualquer outra linguagem. Essa migração pode ser feita aos poucos, já que os módulos e agentes são independentes entre si. Além disso, Perl é portável e os agentes e módulos adicionais ao sistema podem ser desenvolvido em qualquer linguagem.

Para o desenvolvimento do `ida` foram utilizadas também as linguagens de programação Python, *Shell Scripts*, Java e C. Outras tecnologias como o LDAP, usando o gerenciador OpenLDAP, foram utilizadas. Maiores detalhes sobre estas linguagens e tecnologias podem ser encontradas no apêndice C.

Resumidamente, toda a comunicação entre os módulos do `ida` é feita através do banco de dados. Logo, qualquer linguagem que permita acesso ao banco de dados utilizado pode ser usada no desenvolvimento de novos componentes para o sistema.

4.3 Componentes Implementados

A seguir são listados alguns dos componentes implementados no sistema:

Agente de controle: sua função básica é controlar e executar componentes de manutenção que necessitem de um nível maior de acesso a base de dados.

Agentes de manutenção: as funções destes componentes incluem tarefas de controle, verificação e manutenção das informações providas pela base de dados. Entre elas, ajustar as referências cruzadas, converter IPs para nomes de máquina, validar dependências de dados, ajustar dependências de serviços, reciclar chaves de sincronização, manter atualizados os códigos de componentes do sistema e computar chaves de atualizações de sub-árvores de dados.

Agentes de monitoramento: monitorar a atividade e disponibilidade de máquinas e serviços. Os agentes de monitoramento implementados possuem monitores para os serviços IPTables, DNS, DHCP, Apache e MySQL.

Agente local padrão: agente que controla a ativação e execução local dos agentes de serviço. Cada máquina da rede executa um agente local. O agente local desenvolvido permite que quaisquer números ou tipos de agentes de serviços e conseqüentes módulos de geração e atualização sejam atribuídos a máquina dinamicamente.

Módulos de geração e atualização: estão disponíveis módulos de geração e atualização para os seguintes serviços de rede: DNS, DHCP, IPTables, NFS, SSH, MySQL e Apache. Estes módulos são capazes de integrar e automatizar o controle e manutenção desses serviços. Módulos para o controle de máquinas e outros serviços podem ser desenvolvidos e incluídos a qualquer momento no sistema.

Pacotes de *software*: têm como função facilitar o desenvolvimento, controle e manutenção do sistema. Nas seções D.3.0.1, D.3.0.2, D.3.0.3 e D.3.0.4 (do apêndice D) são apresentados maiores detalhes sobre os pacotes de análise de entradas, *log* e temporização.

Ferramentas de uso e manutenção: são ferramentas diversas que auxiliam no gerenciamento geral do sistema. A seção D.3.1 (do apêndice D) lista algumas das ferramentas disponíveis e suas principais funcionalidades.

Maiores informações sobre os componentes implementados podem ser vistas na seção D.2, no apêndice D. Além disso, na seção D.2 também é introduzida a nomenclatura padrão utilizada para os componentes. O objetivo desta padronização foi facilitar a automatização e dinamicidade no processo de manutenção e atualização do código dos componentes do sistema na base de dados.

4.4 Base de Dados

A base de dados é a parte central do `ida`. Todos os componentes ou dependem das informações providas pela base de dados, ou as gerenciam, ou as manipulam, ou as alimentam. É ela quem armazena o código fonte de módulos e agentes, mantém dados sobre dispositivos e serviços da rede, fornece autenticação e políticas de acesso aos seus próprios dados. Devido a isso, a base de dados é de fundamental importância ao sistema.

Para gerenciar a base de dados (utilizando o protocolo LDAP como meio de comunicação e transporte dos dados) foi escolhido o OpenLDAP (OpenLDAP Foundation, 2004a). Este, além de ser livre, tem se mostrado uma boa opção para o gerenciamento de diretórios *online* (THONTON; MUNDY; CHADWICK, 2003; WILLIAMS, 2003; ZEILENGA, 2001a; KUMAR, 2004) e vem sendo suportado, melhorado e utilizado por empresas e corporações como a SUN (Sun Product Documentation, 2004), Apple (Apple Computer, Inc. , 2004; Apple Computer, Inc., 2004), IBM (IBM Linux Technology Center, 2004; KUMAR, 2004), Apsys (Apsys Consultoria Y Sistemas, 2004), SYMAS (BACKES, 2004) entre outras (OpenLDAP Foundation, 2004b).

4.4.1 Interface Padrão de Acesso ao Repositório de Dados

Para facilitar a adaptação do `ida` aos mais diversos contextos administrativos foi criada um interface única de conexão entre os componentes do sistema e a base de dados. Desta forma, a ferramenta não fica restrita a um tipo de protocolo ou base de dados.

Para utilizar o `ida` com um outro sistema de gerenciamento de bases de dados, como o MySQL, DB2 ou Oracle, basta modificar a implementação da interface de comunicação. Ela é composta de alguns métodos básicos que deverão ser redefinidos para a utilização de uma nova base de dados.

A interface mantém uma lista de servidores da base de dados. É o caso de ambientes que desejam alta disponibilidade e confiabilidade do sistema. Conexões de apenas leitura poderão

ser feitas com qualquer um dos servidores. Para que isso seja possível, a base do sistema deverá estar sendo replicada e automaticamente atualizada em todos os servidores. Isso é possível de ser feito através da adequada configuração do OpenLDAP. Com isso, na implementação atual é escolhido aleatoriamente um servidor de conexão para leitura de dados. No caso de escritas, elas são feitas apenas no servidor primário dos dados do sistema.

No caso de uma conexão com a base de dados, caso a conexão com o primeiro servidor falhar será tentada automaticamente uma nova conexão com um outro servidor até que um canal de tráfego de dados seja estabelecido. Essa característica torna o sistema também tolerante a falhas, ou seja, se falhar alguma das máquinas gerentes da base de informações uma das demais, poderá suprir os dados aos componentes do sistema.

Os métodos básicos da interface (pacote `idaNMOSdataBaseAccessObject`) de conexão estão relacionados na seção D.1 do apêndice D.

4.4.2 Sobreposição de Configurações

Na seção 3.4.4 foi definida a sobreposição de dados. Nesta seção são apresentados dois exemplos de sobreposição de dados presentes na implementação atual do sistema.

Dois exemplos da implementação atual de sobreposição de configurações são as descrições dos computadores e dos serviços da rede. Uma máquina pode ter suas configurações específicas como pode também herdar configurações de rede gerais. Caso uma configuração geral precise ser sobreposta, basta definir os atributos desejados na entrada de dados da máquina que o atributos herdados serão sobrepostos. Por exemplo, a configuração geral de rede, onde um determinado conjunto de máquinas herdam as configurações gerais e uma máquina (ou algumas) em específico pertence a uma sub-rede interna distinta. Logo, as configurações desta sub-rede podem ser definidas nas configurações individuais da máquina e serão tomadas como uma sobreposição às configurações gerais do grupo de máquinas. Além disso, uma entrada pode herdar várias descrições gerais ao mesmo tempo. Neste caso, não é garantida a ordem de avaliação e sobreposição das informações das múltiplas heranças. Estas devem ser utilizadas apenas em casos de suprimento de informações complementares.

Os serviços também podem herdar configurações gerais. Com isto é facilitada a tarefa geral de configuração de serviços como o *ssh*, *rlogin* e o *ntp*. Todas as máquinas que possuem estes serviços e precisam ser configuradas podem herdar as configurações gerais do serviços e definir (ou re-definir) apenas as opções ou restrições específicas e particulares do serviço local.

O apêndice B apresenta maiores detalhes sobre a base de dados, relações de dependência/herança e exemplos de entradas cadastradas. Com isso pode-se ter uma idéia prática da aplicação de relacionamento e sobreposição de configurações.

4.5 Relatórios de Atividade do Sistema

Todos os componentes do sistema produzem relatórios do comportamento e execução, que são armazenados na base de dados. Estes relatórios são possíveis devido ao uso do pacote de *log* no desenvolvimento dos componentes do sistema. Erros, alertas, informações crítica e importantes, estado de funcionamento geral e/ou específico e falhas na chamada e execução de rotinas são alguns dos dados registrados pelos componentes do sistema.

Com as informações dos *logs* é possível identificar componentes do sistema que estejam apresentando algum tipo de problema na execução. Sendo assim, tem-se um monitoramento de toda a atividade dos componentes distribuídos do `ida` em um único local.

A seção D.5 do apêndice D apresenta um exemplo de um relatório de atividade.

4.6 Relatórios de Monitoramento de Sistemas e Serviços

Os relatórios de monitoramento podem ser extraídos a partir dos dados armazenados pelos agentes de monitoramento de serviços que fazem parte do domínio administrativo do `ida`. Dados de monitoramento incluem acessibilidade, conectividade e resposta de serviços. Estas informações básicas são suficientes para indicar o estado de disponibilidade e funcionamento de serviços da rede.

A seção D.6 do apêndice D apresenta um exemplo de um relatório de atividade.

4.7 Resumo

Neste capítulo foram apresentadas a arquitetura implementada do `ida`, as tecnologias utilizadas no desenvolvimento e uma abordagem rápida da interface de comunicação. A arquitetura implementada do sistema está funcional e proporciona recursos para o gerenciamento dinâmico, integrado e automatizado de serviços e máquinas de uma rede local de computadores.

Os apêndices D e B apresentam maiores detalhes sobre os componentes atualmente disponíveis no sistema e alguns exemplos de registros de dados da base do sistema implementado. Além disso, são demonstrados alguns exemplos de relação e dependência e explicitado como estes recursos são tratados pelos componentes do sistema.

Somando tudo o que foi apresentado e comentado, o objetivo principal do trabalho foi atingido e o modelo proposto pode ser validado na prática. O próximo capítulo apresenta alguns dados e discussões acerca da validação do sistema.

5 VALIDAÇÃO DO SISTEMA

Este capítulo apresenta alguns cenários de teste e dados estatísticos de execução e avaliação do `ida`. Os dados apresentados têm por objetivo fornecer resultados empíricos e argumentativos que comprovem a funcionalidade e aplicabilidade do sistema.

A próxima seção descreve os ambientes de testes utilizados. Nas seções subseqüentes são apresentados dados de escalabilidade, operação, tolerância a falhas e outras funcionalidades dos `ida`.

5.1 Descrição dos Ambientes de Teste

A rede do Núcleo de Ciência da Computação (NCC), da Universidade Federal de Santa Maria (UFSM), é um exemplo de ambiente que carece de um sistema integrado e automatizado de gerenciamento de serviços e computadores. As soluções existentes resolvem apenas alguns problemas, de forma isolada e independente, dificultando a manutenção, configuração e atualização simultânea de vários sistemas. Esse ambiente foi usado como plataforma de experimentação e validação do sistema proposto neste trabalho.

A figura 5.1 ilustra o cenário utilizado para execução do sistema.

O cenário 1 é composto por um servidor e n máquinas clientes. As máquinas clientes são estações de trabalho dos laboratório do NCC. A tabela 5.1 descreve as configurações dos computadores utilizados. A interconexão é realizada através de equipamentos (comutadores e placas de rede) Fast Ethernet - 100Mbps.

Tabela 5.1: Máquinas utilizadas nos cenários de testes

Tipo	**Identificador**	**Processador**	**Memória**
Servidor	AthlonXP	AthlonXP 1.5GHz	256MB
Cliente	Pentium4	Pentium 4 2GHz	512MB
Cliente	Pentium3c	Pentium 3 Celeron 1GHz	128MB

Todas as máquinas do ambiente de testes contêm a distribuição Gentoo GNU/Linux com o *kernel* em sua versão 2.4, otimizada e compilada para as respectivas arquiteturas em especifico.

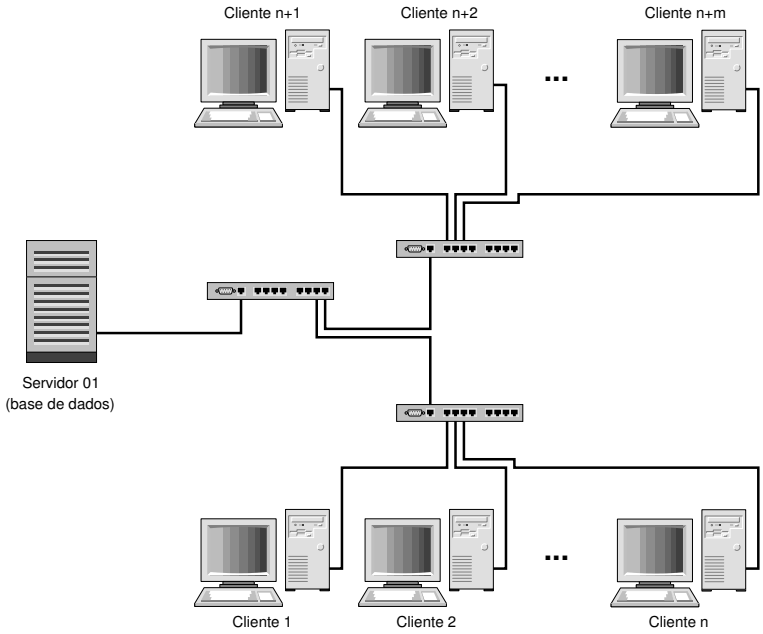

Figura 5.1: Cenário utilizado para os testes

5.1.1 O Servidor

A figura 5.2 ilustra os monitores alocados no servidor dos cenários de teste. Foram inseridos um monitor de tráfego e um monitor de processos.

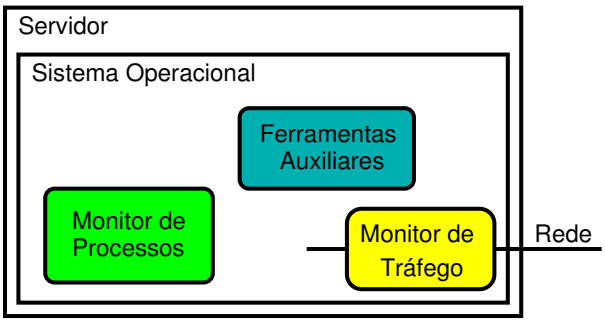

Figura 5.2: Monitores do servidor

Para forçar a execução dos módulos utilizados nos testes, também acompanha o servidor um programa para forçar aleatoriamente as atualizações (execução dos módulos de geração e atualização) e um controlador da freqüência de execução. Este último determina, a cada período definido de tempo, um novo intervalo de execução dos agentes do sistema.

5.1.2 Os Clientes

Nos clientes do ambiente de testes foi instalado um agente local e três agentes de serviços. Cada um dos agentes de serviço possuía um ou mais módulos de geração, atualização ou de execução geral. A figura 5.3 apresenta configuração das máquinas clientes.

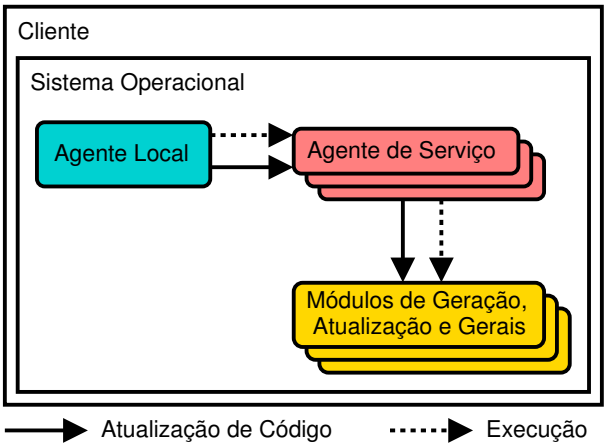

Figura 5.3: Agentes e módulos de um cliente

5.2 Testes de Operação

Componentes básicos como módulos de geração e atualização, agentes de serviços, agentes locais e agentes de manutenção foram testados e postos em execução contínua durante alguns dias. Aparentemente não ocorreram problemas ou erros de execução. As simulações forçavam a execução de componentes do sistema através de uma ferramenta auxiliar, criada especialmente para provocar, em intervalos de tempo aleatórios, incrementos nas chaves de atualização de serviços e componentes. Com isso, serviços e módulos recebiam atualizações de código e módulos de geração e atualização de serviços eram executados.

Durante a segunda fase de testes, com períodos de até três dias sem parar a execução, foram utilizados apenas componentes para a manutenção e atualização dos serviços de DNS e DHCP. Um módulo geral de configuração de máquinas também foi utilizado nestes testes.

As atualizações de dados e o controle de manutenção de códigos fontes foram feitos usando as ferramentas do sistema (ver seção 4.3). Estas apresentaram um bom e correto funcionamento.

Alguns exemplos práticos de atualização automática foram realizados. Um deles levou em consideração a acuracidade de atualização dos serviços de DNS e DHCP. A metodologia e os passos utilizados para o teste foram os seguintes:

1. atribuição dos módulos de geração e atualização do DNS e DHCP a um agente de serviço;
2. atribuição do agente de serviço a um agente local, na máquina em que o serviços deveriam ser mantidos;
3. ativação do agente de verificação e ativação automática de atualizações necessárias;
4. ativação do agente local na máquina servidora dos serviços;
5. escolha de uma máquina qualquer na base de dados;
6. verificação das configurações do servidor DNS e DHCP em relação à máquina escolhida;

7. alteração do endereço IP da máquina escolhida;

8. verificação e confirmação da atualização automática ou não dos dois serviços;

9. contagem do tempo necessário para que as novas configurações fossem detectadas e os serviços fossem adequadamente atualizados.

O objetivo deste teste foi verificar a funcionalidade dos mecanismos de ativação e atualização automática de serviços. O teste realizado ocorreu sem problemas. Após a troca de IP da máquina escolhida os serviços de DNS e DHCP foram corretamente atualizados, de forma automática. A detecção das novas configurações e a respectiva atualização dos serviços levou aproximadamente 20 segundos. O agente de controle de atualizações de dados (responsável pelos dados de ativação dos módulos de geração) estava programado para verificar a existência de novas informações (ou dados atualizados) a cada 10 segundos. O agente local estava programado para executar a cada 20 segundos. Logo, este último tempo explica os aproximados 20 segundos levados para que as atualizações tivessem efeito. Esses tempos podem variar de acordo com as necessidades e requisitos de cada administrador de domínio. Este pode também solicitar uma verificação imediata de novos dados pelo agente de controle de atualizações e uma ativação posterior (também imediata) dos agentes de serviços responsáveis pelos módulos de geração afetados pelas novas informações incluídas na base de dados. Neste contexto, o tempo de detecção e efetivação das atualizações de serviço seria inferior aos 20 segundos levados pela configuração do sistema utilizada para este caso de teste.

5.3 Testes de Tolerância a Falhas

As seções que seguem comentam algumas peculiaridades e cuidados quanto à atualização e migração de serviços, possíveis impactos da re-alocação de serviços vitais e utilização de uma base de dados replicada.

5.3.1 Atualização de Serviços

Os módulos de atualização de serviços e máquinas fornecem precisão e praticidade. Quando executados, estes módulos apenas carregam (da base de dados para o sistema local) os arquivos ou dados de configuração, os copiam para os locais adequados e executam os comandos necessários para que as atualizações tenham efeito. Os módulos de atualização desenvolvidos produzem estes resultados sobre máquinas e serviços como o DNS, DHCP, NFS, SSH e MySQL.

Para configurar cada um dos serviços gerenciados pelo sistema são necessárias informações de descrição dos serviços, dados de mapeamentos, modelos de sobreposição, relações de dependência de dados e configurações gerais. Com excessão das descrições particulares a cada serviço, os demais dados ou relacionamentos são optativos e podem ser utilizados quanto necessário.

No caso dos serviços de DNS, DHCP, NFS, SSH e MySQL são utilizadas informações gerais de cadastro de cada serviço (estes dados serão sobrepostos pelas configurações específicas de cada servidor, quando especificadas), configurações específicas a cada servidor dos serviços (um serviço, como o SSH, pode ser prestado por várias máquinas da rede e possuir restrições particulares a cada uma delas), mapeamentos que indicam os servidores primários e secundários de cada serviço e dependências de dados (roteadores, servidores da rede e máquinas em geral são referenciados pela identificação de suas entradas na base de dados, produzindo consistência e automatização de atualizações). Estes conjuntos de dados e relacionamento permitem uma boa e consistente manutenção e atualização automática dos serviços da rede.

Foram feitos testes de execução e tolerância a falhas. No ambiente de testes, e através de observações, as únicas possíveis falhas dos módulos de atualização foram:

- falha na base de dados (neste caso a atualização não é processada no momento em que o módulo de atualização é executado);
- falha no agente de serviço (neste caso o módulo não é executado, ou é executado parcialmente - caso o agente de serviço for finalizado por algum agente externo durante a execução do módulo de atualização);
- erro na formatação dos dados (neste caso, o módulo de geração não produziu os arquivos de configuração corretos para a configuração do serviço e o módulo de atualização provocou a parada na prestação do serviço após a atualização do mesmo).

Em todos os casos, os erros no funcionamento (ou aplicação de atualizações) dos módulos de atualização foram causados por agentes externos.

5.3.2 Migração de Serviços

As migrações, ou re-alocações, de serviços podem ser realizadas de duas maneiras. Uma delas é através da atuação direta do administrador do sistema. Este retira módulos ou agentes de serviço de um agente local de uma máquina (ou máquina) e os referencia em um agente local de uma outra máquina. O agente local da segunda máquina irá automaticamente detectar os novos componentes sob seu comando e tomar as medidas necessárias para sua ativação e execução. Nos testes realizados, as operações deste gênero foram todas bem sucedidas. Componentes, módulos e agentes de serviço foram alocados de uma máquina para outra sem problemas.

Outra forma de migração de serviço é a automática. Esta deve ser usada com muito cuidado e planejamento. A formação incorreta de políticas de migração e migrações inapropriadas podem levar a problemas e conflitos na prestação de serviços da rede. No entanto, quando programada e utilizada, a migração automática possibilita a ativação dinâmica e rápida de um serviço em uma máquina auxiliar.

Foram realizados alguns testes simples de migração automática de serviços. Um dos testes foi realizado utilizando os serviços de DNS e DHCP. A simulação, em uma rede dedicada, foi

realizada com as seguintes configurações e formas de aplicação:

- foram utilizadas quatro máquinas servidoras, cada uma executando um agente local e um agente de serviço;
- duas máquinas eram servidores primários (responsáveis pelos serviços, um para cada serviço) e as outras duas servidores auxiliares. No caso de falha de prestação de serviço e/ou conectividade os serviços seriam migrados para os servidores auxiliares;
- as quatro máquinas possuíam os mesmos *softwares* instalados, inclusive nas mesmas versões;
- foram simuladas quedas de conectividade com os servidores primários, causando falha na prestação de serviços;
- em caso de falhas de conectividade com a máquina e com o serviço o agente de migração averigua apenas a política de migração, caso presente;
- com a falha de conectividade os agentes de monitoramento do DNS e do DHCP reportaram falha de conexão com a máquina e com o serviço;
- o agente de migração, ao analisar os registros de falha na prestação de serviços, realizou a migração dos módulos de atualização de ambos os serviços;
- o tempo necessário para a migração e configuração dos serviços nas máquinas auxiliares foi de aproximadamente 15 segundos. O agente de migração estava programado para realizar uma verificação dos registros de monitoramento a cada 10 segundos. Os agentes locais estavam programados para executar os agentes de serviço também a cada 10 segundos. Os agentes de monitoramento eram ativados em períodos aleatórios.

A simples experiência comprovou a funcionalidade da migração automática de módulos de configuração de serviços. Os agentes de atualização foram migrados para os agentes de serviço das máquinas auxiliares e quando executados atualizaram as configurações dos serviços locais. Para esse ambiente ser funcional é preciso que os serviços estejam devidamente instalados nas máquinas auxiliares. O tempo de detecção e configuração dos serviços nas máquinas auxiliares pode variar de acordo com a freqüência de tempo de execução dos agentes de monitoramento, do agente de migração e dos agentes de serviços. Estes últimos podem também ser ativados através de uma notificação de ativação imediata aos agentes locais.

Os testes iniciais, realizados sobre módulos do serviço de DHCP e DNS mostraram que se cuidadosamente utilizada a migração automática pode prover um certo grau de liberdade e independência ao sistema, no caso de falhas de serviços e re-estabelecimento de serviços da rede. Todavia, falhas podem ser bastante difíceis de serem detectadas. Uma falha de funcionamento ou prestação de serviço pode ser momentânea e falsa. Alguém pode simplesmente, por alguns

poucos segundos, ter removido um cabo de rede, causando uma quebra de curta duração na conectividade de sistemas. Em um sistema de migração automática, esse curto intervalo de tempo pode ser o suficiente para causar uma migração de serviço. Neste caso, assim que o cabo de rede for reconectado e o antigo serviço primário voltar a responder uma medida de correção, dependendo do serviço, precisaria ser tomada. E existem basicamente duas possibilidades, ou a migração é desativada ou o serviço original é parado. Um dos trabalhos futuros no `ida` é fornecer suporte a ações programadas para esse tipo de situação, utilizando os próprios mecanismos de tolerância a falhas (agentes de manutenção e monitoramento) do sistema.

5.3.3 Possíveis Impactos na Migração de Serviços Vitais

Alguns problemas podem surgir com a migração de serviços vitais a rede. Entre eles o DNS, provavelmente o mais crítico e preocupante. Migrar o serviço de resolução de nomes pode comprometer, temporariamente os acessos de fora para dentro da rede local. Isso por que uma mudança de servidor DNS precisa ser propagada para os níveis superiores da árvore de consultas do sistema de resolução de nomes. Essa propagação pode levar até alguns dias, o que não é algo aceitável mesmo para uma rede de computadores acadêmica. Outra opção seria a troca do IP do servidor. No entanto, a falta de um planejamento detalhado pode levar a inconsistências na rede. Logo, é uma opção para ser utilizada com cautela e preparo.

Para tornar possível e realizável a migração de serviços como o DNS é preciso planejamento e estratégia. Um dos meios mais seguros de realizar esta tarefa é informar de antemão o domínio superior da possível existência de um segundo servidor nomes. Logo, quando o servidor primário falhar os servidores do domínio superior irão re-direcionar suas consultas para o segundo servidor. O serviço de DNS estando ativado e funcionando nesta segunda máquina levará a uma continuidade normal e transparente de funcionamento e acesso a rede local. No entanto, um ambiente desse gênero deve ser preparado com cuidado e previamente testado para evitar problemas futuros.

Outro serviço problemático poderia vir a ser o NFS, pois, normalmente as máquinas do domínio montam estaticamente os diretórios remotos de usuários e sistemas. Este problema poderia ser amenizado através do uso de programas como o *autofs* ou configurações, em arquivos de sistema como o `fstab`, para a montagem dinâmica de sistemas de arquivos de rede. No caso do *autofs* é possível estabelecer que o diretório será montado apenas quando os dados de um determinado diretório estiverem sendo utilizados. Com esta configuração, e utilizando um *alias* para o servidor NFS, os dados remotos seriam dinamicamente montados a cada instante que alguém for utilizá-los, dando a oportunidade de trocar o servidor sem afetar o funcionamento da rede. Entretanto, pontos de montagem em uso provavelmente sofreriam de problemas como o travamento do sistema de arquivos. Neste caso, módulos de execução de comandos poderiam ser utilizados para reiniciar os sistemas de manutenção dos pontos de montagem de sistemas de arquivos remotos.

5.4 Testes de Escalabilidade

Alguns dos testes realizados tiveram como objetivo avaliar aspectos de escalabilidade do sistema, principalmente da base de dados. As próximas seções apresentam dados estatísticos sobre a carga de trabalho do `ida` ou necessidade de recursos em várias configurações. É variado o número de componentes e máquinas controlados pelo sistema e volume e quantidade de dados controláveis pela base de dados.

5.4.1 Módulos e Agentes em Atividade

A seguir serão apresentadas algumas tabelas de dados estatísticos com relação ao número de componentes do sistema. Os dados foram extraídos de um cenário que tinha como objetivo representar casos atípicos de gerenciamento, com períodos períodos de freqüência de execução reduzidos, tentando observar a carga de processamento e dado do servidor do sistema. Além disso, verificar também a estabilidade do sistema com grandes cargas de trabalho.

Alguns detalhes das métricas e seu significado:

intervalo: tempo entre uma execução e outra. Nas simulações realizadas todos os componentes foram programados para levar em conta somente a freqüência de execução global, desconsiderando intervalos de execução particulares ao componente;

uso da CPU: porcentagem da CPU utilizada pelos processos responsáveis pela base de dados;

dados enviados: quantidade média de dados enviados, por segundo, do servidor da base de dados para os componentes do sistema;

dados recebidos: quantidade média de dados enviados, por segundo, pelos componentes do sistema à base de dados. Essas informações foram conseqüentemente armazenadas no servidor do `ida`.

As ferramentas utilizadas para a extração das métricas foram:

top: comando padrão Unix utilizado para monitorar processos do sistema operacional. No caso dos testes realizados, foram monitorados processos de gerenciamento, controle e resposta da base de dados;

ibmonitor: ferramenta de monitoramento e estatística de interfaces de comunicação de uma máquina. Esta foi utilizada para medir a quantidade de dados recebidos e enviados pelo servidor do sistema;

Todas as ferramentas utilizadas para fornecer os dados estatísticos foram configuradas para armazenar métricas a cada intervalo de tempo de um a dois segundos. No caso do *top*, ele foi configurado para extrair apenas dados de uso da CPU dos processos responsáveis pelo atendimento e processamento das requisições feitas à base de dados. Isso por que processos de

manutenção do sistema operacional ou outras aplicações de sistema, como o próprio *top* e os monitores da rede poderiam interferir nas medidas. Em relação ao *ibmonitor*, que monitora o tráfego da rede, não houveram problemas quanto a interferência de outros processos ou aplicações, visto que o servidor estava configurado e reservado apenas para as simulações do `ida`.

Na geração dos dados das tabelas 5.2, 5.3, 5.4 e 5.5 foram utilizados intervalos aleatórios de 0 a 600 segundos[1] para forçar a execução dos módulos (de manutenção de serviços) do sistema. As métricas são médias extraídas de 20 minutos de execução do sistema para cada intervalo de execução.

A execução forçada de módulos, a no máximo cada 10 minutos, foi para verificar alguns limites e carga do sistema. Em uma configuração normal[2] raramente será necessária a atualização de serviços ou máquinas em freqüências tão altas como intervalos de 0 a 600 segundos.

Tabela 5.2: Duas máquinas clientes, com 2 agentes e 2 módulos cada

intervalo (segundos)	uso da CPU (%)	uso máximo da CPU (%)	dados enviados (Kbps)	dados recebidos (Kbps)
2	1.97	5.9	485.28	41.28
4	1.31	2.0	248.40	23.20
8	1.12	2.0	126.24	14.32
16	0.90	2.0	63.84	9.36
32	0.48	2.0	32.64	7.04
64	0.26	2.0	18.48	5.84
128	0.16	2.0	9.04	5.20
256	0.29	7.9	14.40	5.76
512	0.13	2.0	6.56	5.04
1024	0.15	2.0	5.04	4.88

Tabela 5.3: Quatro máquinas clientes, com 2 agentes e 2 módulos cada

intervalo (segundos)	uso da CPU (%)	uso máximo da CPU (%)	dados enviados (Kbps)	dados recebidos (Kbps)
2	2.12	6.9	976.64	88.80
4	1.77	2.0	508.48	52.32
8	1.37	2.0	262.08	33.68
16	1.17	2.0	128.72	17.04
32	0.92	2.0	62.96	9.20
64	0.49	2.0	32.80	6.88
128	0.29	2.0	17.60	5.76
256	0.29	9.8	11.04	5.68
512	0.09	2.0	5.68	4.96
1024	0.08	2.0	4.40	4.88

[1]Este intervalo difere dos intervalos de freqüência de execução presentes nas tabelas. O intervalo aleatório de 0 a 600 segundos é utilizado por uma ferramenta auxiliar para provocar atualizações de dados e conseqüentes execuções dos módulos de geração e atualização dos serviços afetados pelas novas informações.

[2]Em relação a uma rede comum à laboratórios de informática.

Tabela 5.4: Oito máquinas clientes, com 2 agentes e 2 módulos cada

intervalo (segundos)	uso da CPU (%)	uso máximo da CPU (%)	dados enviados (Kbps)	dados recebidos (Kbps)
2	2.22	8.8	1911.04	149.20
4	2.00	2.0	988.08	78.96
8	1.70	2.0	501.12	42.32
16	1.52	2.0	253.12	23.60
32	1.28	2.0	126.24	14.16
64	0.98	2.0	67.28	9.84
128	0.51	2.0	33.12	6.96
256	0.79	11.8	25.52	7.84
512	0.18	3.0	10.72	5.28
1024	0.07	2.0	5.68	4.88

Tabela 5.5: Dezesseis máquinas clientes, com 2 agentes e 2 módulos cada

intervalo (segundos)	uso da CPU (%)	uso máximo da CPU (%)	dados enviados (Kbps)	dados recebidos (Kbps)
2	5.29	32.7	3815.44	293.92
4	2.01	3.0	1966.80	152.64
8	2.01	2.9	1001.04	79.84
16	1.66	2.0	505.76	42.64
32	1.30	2.0	252.32	23.60
64	1.14	2.0	128.72	14.24
128	1.07	2.0	66.08	9.52
256	1.05	13.8	42.24	9.04
512	0.36	2.9	22.64	6.16
1024	0.16	2.0	7.44	5.04

Os ambientes das tabelas 5.2 5.3, 5.4 e 5.5 possuem respectivamente 14, 28, 63 e 112 componentes ativos no sistema. Os dados apresentados são referentes a máquina servidora da base de dados do sistema. A porcentagem de uso da CPU é referente aos 200 maiores medidas (dentro de um universo de aproximadamente 600 medidas) de uso da CPU. Logo, é uma estimativa alta. A média geral fica abaixo deste valor. Mas, como um dos objetivos é justamente apresentar alguns limites de carga, foram utilizados apenas os valores das maiores métricas.

Nas tabela pode ser observado um crescente aumento na porcentagem de uso da CPU e tráfego da rede, a medida que cresce o número de máquinas e conseqüentemente aumenta o número de componentes ativos no sistema. O mesmo acontece, em ordem inversa, com os intervalos de execução. Quanto menor o intervalo, menor o tempo entre uma execução e outra dos componentes do sistema, maior a carga de trabalho da CPU do servidor e maior o tráfego da rede.

Pode-ser perceber, olhando os dados das tabelas, que há uma redução mais amena no tráfego de dados nos intervalos (freqüências de execução) 256, 512 e 1024 segundos. Isso se deve principalmente ao fato de o intervalo aleatório (para forçar a atualização de serviços) ser de 0 a 600 segundos. No caso dos três maiores intervalos de tempo, a probabilidade de executar

atualizações de serviço a cada final de intervalo de tempo é bastante maior em relação aos demais intervalos. Por isso que ocorre uma diferença menos drástica entre a média de dados transmitidos pela rede.

A tabela 5.5, por exemplo, representa 112 componentes ativos no sistema. No caso, foram utilizados apenas componentes para o gerenciamento e manutenção de serviços da rede. Estes serviços, como o DNS e o DHCP, normalmente residem em máquinas servidoras e requerem uma maior carga de dados para serem mantidos/configurados. Dos 112 componentes, 64 são módulos para manutenção de serviços. Estes módulos seriam (na média) o suficiente para gerenciar 32 serviços de uma rede. Como em casos normais a atualização de serviços não ocorre com uma freqüência tão alta quanto a utilizada (0 a 600 segundos, aleatoriamente), o cenário de testes poderia ser encarado como um caso extremo. Em um ambiente normal, tanto o uso da CPU quanto o tráfego na rede seriam menores em relação aos dados apresentados nas tabelas. Levando em consideração os aspectos abordados, pode-se dizer que o sistema não exige um grande poder de processamento e nem consome muita banda de rede em cenários de pequeno e médio[3] porte.

Os testes resultaram em uma base de dados com aproximadamente 600MB de registros de execução e estado dos componentes do sistema. Além disso, o nível de ocupação das CPUs das máquinas clientes, na média (para intervalos pequenos: 2, 4 e 8 segundos), não passou de 2%. Para intervalos maiores esse valor reduz progressivamente.

5.4.2 Volume de Dados

Com o intuito de disponibilizar grandes volumes de dados foram geradas milhões de entradas para serem incluídas no sistema. Cada entrada contém informações simples de identificação, como nome único da entrada e grupo de alocação, e dados fictícios para treze diferentes atributos.

Ao total foram gerados e incluídos na base de dados aproximadamente cinco milhões de registros. Estes registros levaram a uma base de informações com praticamente 7 *gigabytes* de dados armazenados.

Os testes, com grandes volumes de entradas, foram realizados para analisar se a base de dados suportaria um ambiente descomunal de geração e armazenamento de dados.

5.4.2.1 Número Variado de Entradas

A tabela 5.6 apresenta o tempo médio consumido por uma consulta à base de dados, dependendo do número de registros solicitados pela consulta. Uma consulta consiste na conexão com o servidor da base de dados, o processamento da requisição pelo servidor e o retorno e armazenamento dos dados solicitados pelo cliente.

[3] Algumas dezenas de serviços e centenas de máquinas. Os serviços normalmente são concentrados em algumas máquinas. As demais normalmente são apenas estações de trabalho ou máquina cliente, exigindo pouco do sistema de gerenciamento.

Tabela 5.6: Número de entradas e tempo consumido (segundos)

registros	tempo	registros	tempo	registros	tempo	registros	tempo	registros	tempo
10	0.0082	100	0.0310	1000	0.3886	10000	3.8704	100000	254.6475
20	0.0122	200	0.0603	2000	0.6651	20000	5.6566	200000	280.6562
30	0.0149	300	0.0895	3000	0.9979	30000	7.5378	300000	304.6498
40	0.0179	400	0.1198	4000	1.3330	40000	9.7690	400000	332.3401
50	0.0219	500	0.1489	5000	1.6680	50000	12.3060	500000	357.2332
60	0.0238	600	0.1783	6000	1.9918	60000	14.8506	600000	380.3561

Estes dados dão uma idéia do desempenho e suporte da base de dados sobre a interface do OpenLDAP. Neste caso, cada entrada contém em torno de 500 *bytes* de dados. O objetivo não foi testar a quantidade de dados mas sim o impacto do número de entradas sobre a consulta, principalmente em relação ao tempo de processamento consumido pelo servidor do sistema. A maior parte do tempo das consultas maiores é devido ao processamento necessário no servidor de dados para completar as requisições de dados.

5.4.2.2 Entradas de Tamanhos Diversos

As tabelas 5.7 e 5.8 apresentam alguns dados quanto ao tempo de inclusão e consulta para entradas com tamanhos diversos. Cada consulta ou inclusão refere-se a média de 100 operações sobre 10 entradas de um determinado tamanho. Neste contexto, os valores das tabelas representam o tempo médio de consulta ou inclusão de uma entrada.

A tabela 5.7 mostra alguns dados referentes a entradas de dados com tamanhos variados. A tabela 5.8 é uma seqüência da tabela 5.7, apresentando entradas de dados (em *kbytes*) com tamanhos maiores.

Tabela 5.7: Tempo de inclusão e consulta

operação	**tamanho das entradas (em Bytes)** e tempo (em milisegundos)										
	4	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**	**2048**	**4096**
inclusão	3.9	3.9	3.9	3.9	3.9	4.0	4.0	4.1	4.4	4.9	4.9
consulta	4.2	4.2	4.2	4.1	4.2	4.1	4.2	0.8	0.8	0.9	0.9

Tabela 5.8: Tempo de inclusão e consulta

operação	**tamanho das entradas (em kbytes)** e tempo (em milisegundos)										
	4	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**	**2048**	**4096**
inclusão	4.9	5.5	6.6	8.5	12.2	23.3	39.7	85.6	116.9	304.6	331.1
consulta	0.9	1.1	1.3	1.8	4.0	5.7	10.9	22.7	46.4	91.9	182.5

Observando os dados apresentados pode-se dizer que o tempo de inclusão na base de dados costuma ser maior, aumentando de acordo com a quantidade de dados a ser armazenada. Isso é uma característica típica de grande parte dos sistemas de gerenciamento de banco de

dados. O motivo é que em muitos casos os usuários necessitam de basicamente bons tempos de consulta. Normalmente, a maior atividade em bancos de dados é na consulta de dados. As inclusões costumam ser menos freqüentes. No `ida` é semelhante, com a excessão dos registros de monitoramento e atividade de serviços da rede e do próprio sistema.

Nos casos de teste, as entradas de registro não passaram de 2 *kbytes* de dados. Considerando que o tempo de inclusão de uma entrada de quatro *kbytes* de dados é de aproximadamente 4.9 milisegundos, em torno de 200 registros de *log* podem ser gerados por segundo.

Durante os testes foi constatado que o limite máximo de valor de um atributo é de aproximadamente 4MB de dados. Porém, as entradas da base de dados suportam até 12MB de dados. Esses limites, a princípio, não tem nenhum impacto sobre o funcionamento do sistema. Dificilmente um ambiente administrativo necessitará do uso de atributos com grandes quantidades de dados ou mesmo entradas com vários *megabytes* de dados.

5.5 Resumo

Este capítulo abordou alguns aspectos de funcionamento e aplicação prática do `ida`. Os testes de operação e tolerância a falhas mostraram que o sistema é aplicável e funcional para o gerenciamento de serviços e máquinas de redes locais. Alguns de seus recursos fornecem flexibilidade e automaticidade no gerenciamento de máquinas e serviços, facilitando a vida de administradores de sistemas.

Os dados sobre escalabilidade mostraram que o sistema é aplicável a uma boa variedade de ambientes administrativos, visto que a maioria não precisa de suporte a volumes exorbitantes de dados (em espaços curtos de tempo) e nem quantidades muito grande de componentes (agentes e módulos).

O próximo capítulo apresenta a conclusão e os trabalhos futuros. Estes visam incrementar as funcionalidades e contextos de aplicação do `ida`.

6 CONCLUSÃO E TRABALHOS FUTUROS

6.1 Conclusão

O gerenciamento de redes de computadores é uma área em plena expansão. Grandes esforços estão sendo realizados para formar e capacitar pesquisadores e técnicos de alto nível para o gerenciamento de redes de computadores.

Ao mesmo tempo, em uma esfera global, pesquisas e necessidades de domínios administrativos indicam uma carência no que diz respeito a ferramentas integradas para o gerenciamento de máquinas e serviços. É com o intuito de suprir algumas destas demandas que surgiu a proposta deste trabalho.

Neste contexto, este trabalho levantou características importantes para sistemas de gerenciamento integrado de redes de computadores e apresentou o projeto e desenvolvimento de um sistema capaz de amenizar várias das deficiências de outras ferramentas de gerenciamento. O resultado é um protótipo de um sistema de gerenciamento integrado e automatizado de serviços e computadores de redes locais, denominado `ida`. Este sistema possui grande parte das características consideradas importantes em uma ferramenta que pretenda abranger o maior número possível de domínios administrativos e situações do dia-a-dia de administradores de redes, bem com, prover mecanismos simples e úteis para o gerenciamento integrado e automatizado dos mais diversos serviços e dispositivos de uma rede de computadores.

Outro aspecto importante é o fato de o `ida` fornecer uma arquitetura e abstração que permitem a integração de sistemas de gerenciamento de redes existentes, utilizando seus recursos e especialidades na construção de um ambiente de controle e gerenciamento mais completo.

O projeto, desenvolvimento e validação do sistema resultaram no cumprimento dos objetivos propostos pelo trabalho. Além disso, abriram a possibilidade para projetos e planos de continuidade na evolução e abrangência do `ida`. Neste sentido, os trabalhos futuros (ver capítulo 6.2) também terão uma contribuição significativa na expansão do sistema.

Os resultados de validação do sistema demonstraram que ele é aplicável em ambientes reais e constitui uma boa ferramenta de gerenciamento para contextos administrativos diversos. Sua aplicabilidade vai desde simples redes locais, que lidam apenas com computadores e impressoras, até redes mais complexas, que precisam manipular e controlar dispositivos variados

como pontos de acesso de redes sem fio, computadores de mão (PALMs, PDAs), *firewalls* e roteadores, ou ainda ambientes administrativos que necessitem de funcionalidades e recursos específicos.

6.2 Trabalhos Futuros

As próximas seções apresentam alguns dos trabalhos que podem ser realizados para melhorar o `ida`, aumentar suas funcionalidades e aplicações e integrá-lo com outros sistemas de gerenciamento de redes. As seções que seguem apresentam novos recursos e componentes que poderiam ser implementados ou incorporados no `ida`, possíveis integrações com outros sistemas e mecanismos de gerenciamento de serviços de uma rede e outros trabalhos e aplicações do sistema.

6.2.1 Novos Recursos

6.2.1.1 Sistema de Auto-diagnóstico

O `ida` é provido de um sistema de *logs* que possibilita que todos os seus componentes armazenem seus registros de execução em uma sub-árvore da base de dados. Existe ainda uma sub-árvore destinada ao monitoramento do tempo gasto por cada componente na realização de tarefas como consultas a bases de dados, inclusão de dados, modificação de dados, remoção de dados, escrita em arquivo, *download* de pacotes de classes necessários para o funcionamento de agentes e módulos, entre outras funcionalidades.

Os conjuntos de informações acerca dos componentes do próprio `ida` fornecem dados que expressam a eficiência e o estado atual de funcionamento de componentes do sistema, locais ou distribuídos. O sistema é acompanhado de ferramentas que permitem uma visualização rápida e simples desses dados (ver seção D.5 do apêndice D e seção 4.5 do capítulo 4). Seria interessante desenvolver um módulo que faça a análise dessas informações e seja capaz de fornecer visões gerais de funcionamento e identificar problemas de execução dos componentes do sistema de forma automática e dinâmica. O objetivo é encontrar e diagnosticar automaticamente falhas ou problemas gerais em componentes do `ida`.

Esse módulo poderia também possuir funcionalidade básicas de recuperação e resolução de problemas, como a reconfiguração, re-execução, atualização, migração ou remoção de componentes problemáticos.

6.2.1.2 Adaptação de uma Interface Gráfica

Outro trabalho futuro é a criação de uma interface gráfica específica aos requisitos e necessidades do `ida`. Esta poderia fornecer direções e ajuda ao usuário na hora de gerenciar a base de dados ou o sistema como um todo. Além disso, poderiam validar as configurações e relações de dependência em tempo de edição.

6.2.1.3 Implementar Classes em Diferentes Linguagens

Alguns pacotes de *software* foram criados para facilitar o desenvolvimento e manutenção do `ida`. As mesmas classes, como `idaDataBaseConnection`, `idaTimingObject`, `idaLogReportObject` e o pacote `idaGeneralFunctions`, poderiam ser implementadas em outras linguagens de programação, como Java, C e Python. Estes pacotes fornecem recursos úteis e práticos para o desenvolvimento, monitoramento e gerenciamento do `ida`. Logo, disponibilizá-los em outras linguagens facilitaria o desenvolvimento ou adaptação de componentes do `ida` por usuários que gostariam de utilizar sua linguagem preferida, e os recursos destes pacotes.

6.2.1.4 Sistema de Registro de Alterações

Com a implementação de um sistema transparente para o registro e manutenção de históricos de alterações feitas no sistema, tanto por administradores quanto por componentes do `ida`, seria possível a recuperação de estados de configuração consistentes do sistema. Esse recurso evitaria perdas por descuidos e falhas de administração.

6.2.1.5 Descrição de Dispositivos de Rede em Geral

Para simplificar e padronizar o cadastro, controle e gerenciamento de dispositivos de rede como impressoras, HUBs, comutadores, PDAs e roteadores, poderia-se criar modelos e exemplos de descrição de dispositivos de rede no geral. Isso inclui definir uma classe LDAP para dispositivos em geral e a criação de uma nova sub-árvore de dados.

6.2.2 Novos Componentes

6.2.2.1 Permitir Alterações Manuais Locais

Para permitir que alterações de configuração locais possam ser feitas, seria interessante desenvolver um componente auxiliar que monitore e controle os processos de modificação local. Após as alterações tenham sido processadas localmente, elas devem ser reportadas transparente e automaticamente a base de dados. Com isso, o sistema torna-se mais interessante em domínios administrativos com múltiplos gerentes de sistemas, que muitas vezes gostam de realizar alterações emergências manualmente.

6.2.2.2 Módulos de Gerenciamento de Agentes SNMP

SNMP é um protocolo de gerenciamento largamente difundido. Equipamentos de redes normalmente possuem suporte a agentes SNMP.

Para possibilitar o gerenciamento de serviços e sistemas também através do SNMP, um dos objetivos futuros é desenvolver módulos genéricos de controle e gerenciamento SNMP que guardem e busquem seus comandos e dados na base de dados do `ida`. Isso possibilitará um maior grau de compatibilidade com sistemas de gerenciamento SNMP disponíveis e utilizados

por administradores de rede.

6.2.2.3 Módulos de Integração com o Cfengine e o LCFG

Ferramentas como o Cfengine e o LCFG são úteis e práticas para a realização de uma série de configurações de serviços e sistemas. No entanto, são linguagens específicas e restritas, não apropriadas para usos mais gerais ou representações mais abstratas de dados.

Entre as operações básicas que estas ferramentas são capazes de executar estão:

- verificação, criação e remoção de *links*;
- cópias simples de arquivos;
- controle de pontos de montagem;
- controle centralizado de agendamento de tarefas (estilo o *cron*);

Devido a isso, o desenvolvimento de componentes para o gerenciamento e configuração desse tipo de sistemas seria de bom proveito ao `ida`. Tarefas corriqueiras de manutenção poderiam ser configuradas e mantidas por ferramentas como o Cfengine e o LCFG.

Adicionalmente, o Cfengine, por exemplo, poderia ser utilizado como um substituto a alguns módulos de atualização de serviços e máquinas. Isso poderia acarretar em configurações mais complexas e exigir maiores cuidados por parte do administrador do sistema. Entretanto, com um módulo para a configuração automática do Cfengine essa tarefa seria amenizada e as descrições de configuração e gerenciamento poderiam ser feitas em um nível mais abstrato e geral.

Além disso, com as configurações do Cfengine e do LCFG sendo mantidas pelo `ida`, relações mais abstratas e dinâmicas de dependência (entre essas ferramenta e os demais componentes do sistema) e atualização de dados poderiam ser mantidas.

6.2.2.4 Módulo genérico (alto nível) de controle de permissões

Uma opção em relação ao uso associado de sistemas como o Cfengine e o LCFG é o desenvolvimento de módulos genéricos e flexíveis para tarefas específicas como o monitoramento de arquivos e sistemas de arquivos. A vantagem esta no desenvolvimento de módulos flexíveis e extensíveis, a caráter e necessidade do administrador do sistema.

Um módulo específico poderia ser capaz de gerar alertas e apontar falhas aos componentes de monitoramento do sistema, levando a investigação e diagnósticos rápidos.

6.2.2.5 Módulos Específicos

Um exemplo de um módulo específico poderia ser um componente que seja capaz de realizar a troca dinâmica de endereços IP de conjuntos de máquinas. Essa troca deve ser realizada de forma ordenada e sincronizada, para evitar a ocorrência de conflitos de IP.

Um componente de troca dinâmica de conjuntos de IPs é especialmente útil para a criação de sub-redes internas específicas ou provisórias. Estas podem ser úteis para o teste e análise

de uma série de sistemas e ferramentas. O que é especialmente interessante em laboratórios e centros de pesquisa em computação.

6.2.2.6 Gerenciamento de Instalações

Um incremento a extensão do `ida` seria a criação de módulos especializados na instalação de *software* em geral. Com estes componentes seria possível instalar e manter simultaneamente uma boa diversidade e quantidade de sistemas.

Os componentes de instalação proveriam uma interface distribuída para o gerenciamento e ativação de instalações completas ou parciais de máquinas e dispositivos de rede. Uma meio simples de prover um mecanismo para o gerenciamento parcial de instalações é através de uma interface (módulo) que forneça dados e operações a aplicativos como o CheckInstall (DURáN, 2004), que possibilita através de uma única interface a instalação de pacotes de *software* em múltiplas distribuições GNU/Linux.

6.2.3 Integração de Sistemas

6.2.3.1 Integração com Sistemas de Monitoramento

O `ida` fornece uma arquitetura para o monitoramento integrado e dinâmico de dispositivos e serviços de rede. Alguns componentes específicos de monitoramento foram desenvolvidos e estão presentes no sistema. Outros componentes de monitoramento de sistemas ainda são necessários. Existem boas ferramentas de monitoramento de dispositivos de rede, como o FreeNMS, o Big Brother e o SNMPMonitor. Os recursos destes sistemas poderia ser integrado ao `ida` através do desenvolvimento de interfaces de integração, fornecendo um ambiente prático e útil para o monitoramento de sistemas, diagnóstico de falhas e tolerância a falhas (previsão de falhas).

6.2.3.2 Integração com Sistemas de Controle de Usuários

Sistemas de autenticação de usuários são alguns dos recursos praticamente sempre presentes em redes de computadores. Estes sistemas podem ser baseados em arquivos de sistema ou em diretórios *online*. Sua função básica é fornecer um meio de validar e autenticar usuários em uma máquina, programa ou na rede.

Com isso, busca-se integrar o gerenciamento de usuários ao `ida`. Como este utiliza o LDAP, esta integração fica relativamente fácil. Além disso, hoje muitos domínios administrativos estão utilizando ou migrando seus sistemas de autenticação para o LDAP.

Com este novo conjunto de informações agregadas ao `ida` será possível um controle integrado de acesso a roteadores, servidores squid, entre outros sistemas que desejem realizar autenticação em nível de usuário. Isso poderá ser feito estaticamente, atualizando automaticamente arquivos de configuração de serviços a partir dos usuários cadastrados no sistema ou através da autenticação direta com o diretório de dados do LDAP.

Outra aplicação é o próprio controle e criação de contas de usuários para a rede local. Com o

desenvolvimento de um componente especializado em cadastro e controle de contas de usuários tarefas como criação do diretório de dados, e-mail de boas vindas, senha aleatória e até o *login* do usuário poderiam ser gerados de forma automática e dinâmica. O administrador do sistema precisaria apenas cadastrar o usuário na base de dados.

A criação de componentes para o gerenciamento de usuários de uma rede de computadores será um passo a mais na integração de sistemas gerenciáveis pelo `ida`.

6.2.3.3 Integração com Sistemas de Predição de Falhas

Um dos métodos para assegurar a estabilidade da rede é através da descoberta e resolução do maior número possível de falhas na rede. Esse é o ponto onde sistemas de gerenciamento de falhas têm um importante papel em redes de computadores (MURDOCH, 2003).

Com a evolução e expansão das redes de computadores existe uma crescente necessidade de sistemas que sejam capazes de manipular erros. Algumas pesquisas estão apresentando opções como *proxies* para melhorar a dependabilidade de sistemas de gerenciamento de falhas (JR et al., 1998).

Neste contexto, esta proposta de trabalho futuro é agregar ao `ida` ferramentas de predição e análise de falhas e problemas na rede em geral, com o objetivo de desenvolver uma interface que seja capaz de transformar os dados de monitoramento (armazenados pelos componentes de monitoramento agregados ao `ida`) em informações formatadas para ferramentas de predição e análise de falhas (NUNES, 2003; NUNES et al., 2004). A partir dos dados processados, esse sistema poderia permitir previsões e medidas que sejam capazes de evitar problemas como indisponibilidade de serviços e congestionamento da rede.

6.2.3.4 Segurança Coordenada e Integrada

Um dos grandes problemas em redes de computadores é a segurança de dados, sistemas e usuários. Com a expansão das redes fica cada vez mais difícil e custoso o gerenciamento de sistemas de segurança. É comum existirem redes com vários *firewalls* (internos e externos), sistemas locais e individuais para estabelecer segurança a máquinas ou sistemas específicos e mecanismos gerais de monitoramento e segurança da rede.

Os ambientes de segurança estão ficando cada vez mais complexos e difíceis de serem gerenciados. Com a criação de interfaces de comunicação e troca de informação padrãos entre sistemas de segurança e o `ida`, seria possível simplificar e integrar o gerenciamento de diferentes sistemas de segurança. Isso permitiria que esses sistemas fossem gerenciados e configurados automaticamente através dos recursos do `ida`.

Com uma boa arquitetura de controle, troca e compartilhamento de informações espera-se que seja possível estabelecer níveis de segurança mais abrangentes onde ataques, falhas e problemas de segurança cheguem rapidamente ao conhecimento de todos os sub-sistemas de segurança da rede. Neste ambiente, um ataque poderia ser mais rapidamente detectado e barrado.

O `ida` fornece componentes desenvolvidos e uma arquitetura apropriada para o projeto,

desenvolvimento e implantação de um sistema integrado de segurança no nível anteriormente exposto. Além disso, com uma organização compartilhada e integrada de segurança será possível a implementação de um coordenador geral de segurança. Este teria o papel de comandar e alertar os demais sistemas de segurança da rede. Procedimentos automáticos ou manuais poderiam rapidamente serem solicitados a esse coordenador central. Um exemplo poderia vir a ser uma nova, emergencial e temporária política de monitoramento e controle de tráfego, onde sistemas como *proxies*, *sniffers* e filtros de pacotes seriam comandados a realizar um monitoramento e controle de pacotes e conexões com características e padrões estabelecidos pelo administrador do sistema.

6.2.4 Outros Trabalhos e Aplicações

6.2.4.1 Gerenciamento de Outros Dispositivos de Rede

Atualmente o `ida` dispõe de recursos para o gerenciamento de estações de trabalho, servidores e computadores clientes de um modo geral. Um objetivo futuro factível a curto e médio prazo é o desenvolvimento de componentes para o gerenciamento de dispositivos como impressoras, chaveadores, roteadores e outros dispositivos de rede gerenciáveis ou que sejam programáveis e atualizáveis. o `ida` poderia controlar regras de acesso, configurações gerais, políticas de conexão (como filtros de pacotes), atualização de *software*, entre outras aplicações.

6.2.4.2 OpenLDAP com Outros Gerenciadores de Dados

OpenLDAP é uma ferramenta com código e arquitetura aberta para o gerenciamento de diretórios no formato do protocolo LDAP. Em sua versão atual suporta os gerenciadores de bases de dados LDBM e BDB. Possui uma API definida e que permite a utilização de qualquer sistema de gerenciamento de banco de dados.

Uma das idéias é testar o OpenLDAP e sua escalabilidade sobre sistemas de banco de dados como o Interbase, Firebird e MySQL. Os dois últimos são *softwares* livres e estão cada vez mais sendo utilizados e desenvolvidos pela comunidade acadêmica e corporativa.

Sistemas como o Firebird e MySQL possuem teoricamente mais recursos de gerenciamento e uma escalabilidade e controle melhor sobre os dados em relação ao LDBM e o BDB. Para redes de grande porte (onde talvez GigaBytes de dados diários tenham que ser armazenados e gerenciados) um bom suporte do banco de dados por trás do OpenLDAP será fundamental para o `ida`. Escalabilidade, desempenho, gerenciamento seguro de transações, consistência e integridade são pontos cruciais para a manutenção da base de informações, dados e códigos do sistema.

6.2.4.3 Utilização Direta de um Banco de Dados pelo `ida`

Outra opção a ser implementada, testada e validada é a utilização direta de um sistema de gerenciamento de banco de dados. Neste caso, sem o uso do OpenLDAP e do protocolo de comunicação LDAP.

Para tanto será necessário uma análise e configuração apropriada da base de dados ou o desenvolvimento de um sub-sistema intermediário que ficará responsável pelo gerenciamento de conexões, autenticação e autorização dos componentes do `ida`. Uma interface simples e específica ao `ida` pode ser uma boa opção para comparar características de desempenho, gerenciamento e eficiência contra o sistema utilizando o protocolo LDAP e gerenciador de dados OpenLDAP. Outros aspectos como sobrecarga de processamento e tráfego das comunicações seriam pontos passíveis de avaliação e comparação.

6.2.4.4 Representação XML

XML é uma tecnologia de representação de dados que vem sendo utilizada cada vez mais no gerenciamento de redes (LEE; CHOI, 2002; ADWANKAR, 2004; ANDERSON; SCOBIE, 2002).

Para manter compatibilidade e poder usufruir dos recursos e vantagens do XML é interessante o desenvolvimento de uma módulo que seja capaz de representar e compreender XML. No caso do LDAP, conversões LDAP para XML e XML para LDAP são atualmente possíveis de serem feitas. Um exemplo é a ferramenta XML Data Mediator da IBM (IBM alphaWorks, 2002). Esta ferramenta realiza a conversão no sentido LDAP=>XML e XML=>LDAP, ou seja, pode ser útil ao `ida`.

Com um módulo geral de geração de conversão LDAP <=> XML seria possível adaptar o `ida` para o controle e configuração de protocolos de gerenciamento de dispositivos de rede como o NetConf.

6.2.5 Resumo

O `ida` ainda está em fase inicial de desenvolvimento, apesar de poder ser utilizado na prática para o gerenciamento e operação de serviços e sistemas de redes em geral. Os trabalhos futuros apresentados neste capítulo mostram alguns exemplos de possibilidades para estender e melhorar o sistema.

Muitos destes trabalhos futuros tornarão o `ida` uma ferramenta mais completa e aplicável a uma gama maior de domínios administrativos. Enfim, todas estas contribuições virão a incrementar suas possibilidades e aplicabilidades no gerenciamento de quaisquer redes de computadores e dispositivos em geral.

REFERÊNCIAS

IDEALX Contributions to the Samba project. **SMBLDAP tools**. http://samba.idealx.org/index.en.html. Último acesso realizado em março de 2004.

ABBEY, J. **Ganymede**. jonabbey@arlut.utexas.edu, http://tools.arlut.utexas.edu/gash2/. Último acesso realizado em dezembro de 2004.

ABBEY, J.; MULVANEY, M. Ganymede: an extensible and customizable directory management framework. In: CONFERENCE ON SYSTEMS ADMINISTRATION (LISA-98), 12., 1998, Berkeley, CA. **Proceedings...** USENIX Association, 1998. p.197–218.

Active Working Group. **Network Configuration (netconf)**. http://custom.lab.unb.br/pub/rfc/ietf/netconf/netconf-charter.txt, http://www.ietf.org/html.charters/netconf-charter.html, http://www.ietf.org/ietf/netconf/netconf-charter.txt. Último acesso realizado em dezembro de 2004.

ADHICANDRA, I.; PATTINSON, C.; SHAGHOUCI, E. Using Mobile Agents to Improve Performance of Network Management Operations. In: POSGRADUATE NETWORKING CONFERENCE, 2003, Liverpool, United kindgom. **Anais...** [S.l.: s.n.], 2003.

AdRem Software, Inc. **AdRem NetCrunch User's Guide**. [S.l.: s.n.], 2004. http://www.adremsoft.com. Último acesso realizado em dezembro de 2004.

ADWANKAR, S. **NetConf Data Model**. [S.l.: s.n.], 2004. Internet Draft. http://www.rfc-editor.org/internet-drafts/draft-adwankar-netconf-datamodel-01.txt. Último acesso realizado em dezembro de 2004.

alan@akbkhome.com. **Welcome to the new revamped LDAP Schema Viewer**. http://ldap.akbkhome.com/. Último acesso realizado em novembro de 2004.

ALBERT, T. **Technical Communicators and Scripting: why tcl, perl, and shell scripting matter to us**. http://www.wordesign.com/linux/scripting-for-tech-com.htm. *Summary and synthesis of the Silicon Valley Linux User's Group Meeting*. Último acesso realizado em dezembro de 2004.

Alchemy Lab. **NetMonitor - Alchemy Eye**. http://www.alchemy-lab.com/products/eye/. Último acesso realizado em dezembro de 2004.

AMADOR, M. **Directory administrator**. http://diradmin.open-it.org/. Último acesso realizado em dezembro de 2004.

ANDERSON, P. Towards a High-Level Machine Configuration System. In: USENIX CONFERENCE ON LARGE INSTALLATION SYSTEM ADMINISTRATION (LISA), 8., 1994, Berkeley, CA. **Anais...** USENIX Association, 1994. p."19–26".

ANDERSON, P. **The Complete Guide to LCFG** . 2001.

ANDERSON, P. **A Declarative Approach to the Specification of Large-Scale System Configurations**. [S.l.: s.n.], 2001. DICE Computing Environment Project, Division of Informatics, University of Edinburgh. http://homepages.inf.ed.ac.uk/dcspaul/publications/conflang.pdf. Baixado em novembro de 2004.

ANDERSON, P.; GOLDSACK, P.; PATERSON, J. SmartFrog meets LCFG - Autonomous Reconfiguration with Central Policy Control. In: LARGE INSTALLATIONS SYSTEMS ADMINISTRATION (LISA) CONFERENCE, 2002., 2003, Berkeley, CA. **Proceedings...** [S.l.: s.n.], 2003.

ANDERSON, P.; SCOBIE, A. LCFG: the Next Generation. In: UKUUG WINTER CONFERENCE, 2002. **Anais...** [S.l.: s.n.], 2002. http://www.lcfg.org/doc/ukuug2002.pdf. Baixado em outubro de 2004.

ANGUISH, S.; BUCK, E. M.; YACKTMAN, D. A. **Cocoa Programming**. [S.l.]: Sams Publishing, 2003.

Apple Computer, Inc. . **Mac OS X open at the source.** http://www.apple.com/opensource/. Último acesso realizado em dezembro de 2004.

Apple Computer, Inc. **Mac OS X Server: basic introduction to netinfo domains**. http://docs.info.apple.com/article.html?artnum=30832. Último acesso realizado em fevereiro de 2004.

Apple Computer, Inc. **Mac OS X Server 1.x: what is netinfo?** http://docs.info.apple.com/article.html?artnum=60038. Último acesso realizado em fevereiro de 2004.

Apple Computer, Inc. **Understanding and Using NetInfo**. [S.l.: s.n.], 2001. http://manuals.info.apple.com/Apple_Support_Area/Manuals/software/Under%standingUsingNetInfo.PDF. Baixado em dezembro de 2004.

Apple Computer, Inc. **Mac OS X Server v10.3 introduces Open Directory 2, the latest version of Apples standards-based directory and authentication services architecture**. http://www.apple.com/server/macosx/open_directory.html. Último acesso realizado em dezembro de 2004.

Apsys Consultoria Y Sistemas. **Networking e Infraestrutura**. http://www.apsys.com.br/ACM-Apsys-por.nsf/Content/SE-Infrap. Último acesso realizado em dezembro de 2004.

Arusha Project Team. **The Arusha Project home page**. http://ark.sourceforge.net/. Último acesso realizado em dezembro de 2004.

ATKINSON, L. **Core PHP Programming**. 3.ed. [S.l.]: Prentice Hall, 2003.

AUGERAT, P. **Ka Clustering Tools**. http://ka-tools.sourceforge.net/. Último acesso realizado em outubro de 2004.

AUGERAT, P.; BILLOT, W.; DERR, S.; MARTIN, C. A scalable file distribution and operating system installation toolkit for clusters. In: CCRID 2002, 2002. **Proceedings...** [S.l.: s.n.], 2002.

AZAMBUJA, M. C. de. **PSWeM: desenvolvimento e implementação de uma ferramenta baseada na web para gerenciamento de redes ao nível de serviço**. 2001. 110 páginasp. Dissertação de mestrado em Engenharia Elétrica — Pontifícia Universidade Católica do Rio Grande do Sul — PUCRS, Porto Alegre, Brasil.

BACKES, M. **Directory Solutions Using OpenLDAP Overlays**. http://www.symas.com/techtips/introtooverlays.html. Último acesso realizado em dezembro de 2004.

BAO, W. Introduction to Security of LDAP Directory Services. **SANS Security Essentials**, [S.l.], May 2001. http://www.giac.org/practical/gsec/Wenling_Bao_GSEC.pdf. Baixado em julho de 2004.

BEAZLEY, D. M. **Advanced Python Programming**. Slides of O'Reilly Open Source Conference.

BETTS, C. **JXplorer - Java LDAP Browser**. http://pegacat.com/jxplorer/. Último acesso realizado em novembro de 2004.

BIALASKI, T. **Directory Server Security**. [S.l.: s.n.], 2000. http://www.sun.com/blueprints. Último acesso realizado em dezembro de 2004.

BIESZCZAD, A.; WHITE, T.; PAGUREK, B. Mobile Agents for Network Management. **IEEE Communications Surveys**, [S.l.], 1998.

Borland. **Delphi Language Guide**. 100 Enterprise Way, Scotts Valley, CA 95066-3249: Borland Software Corporation, 2002. http://www.borland.com.

BRADNER, S. **Key words for use in RFCs to Indicate Requirement Levels**. [S.l.: s.n.], 1997. RFC 2119. Category: Standards Track. http://www.faqs.org/rfcs/rfc2119.html.

BUENO, A.; CARISSIMI, A. Análise Comparativa entre NIS e LDAP. In: SEGUNDA ESCOLA REGIONAL DE COMPUTADORES, 2004. **Anais...** [S.l.: s.n.], 2004. p.89–94.

BURGESS, M. A Site Configuration Engine. In: USENIX COMPUTING SYSTEMS, 1995, Norway. **Anais...** [S.l.: s.n.], 1995.

BURGESS, M. **Cfengine Concepts**. Norway: Faculty of Engineering, Oslo University College, 2001.

BURGESS, M. **Cfengine Reference**. Norway: Faculty of Engineering, Oslo University College, 2004.

BURGESS, M.; CANRIGHT, G. Scalability of Peer Configuration Management in Partially Reliable and Ad Hoc Networks. In: INTEGRATED NETWORK MANAGEMENT, 2003. **Anais...** [S.l.: s.n.], 2003. p.293–305.

BURKE, C. **Kylix/Delphi LDAP Support**. http://codecentral.borland.com/codecentral/ccweb.exe/listing?id=16879. Último acesso realizado em dezembro de 2004.

BURKE, S. M. **Perl & LWP**. [S.l.]: O'Reilly & Associates, 2002. Fetching Web Pages, Parsing HTML, Writing Spiders & More.

CAMERON, J. **Webmin**. www.webmin.com. Último acesso realizado em dezembro de 2004.

CANTù, M. Internet Programming with Delphi. **Borland Developer Network**, [S.l.], 2001. http://community.borland.com/article/0,1410,27143,00.html. Último acesso realizado em dezembro de 2004.

CASE, J.; FEDOR, M.; SCHOFFSTALL, M.; DAVIN, J. **Simple Network Management Protocol (SNMP)**. [S.l.: s.n.], 1990. RFC 1157. http://www.faqs.org/rfcs/rfc1157.html. Último acesso realizado em dezembro de 2004.

CASSAR, J.-P.; KAHLON, K.; NYHOF, S.; SEGERS, R. **Integrated Centralized Automated/Advanced Operation**. 1.ed. [S.l.]: IBM RedBooks. IBM International Technical Support Organization, 1996. 320p.

CASTAN, D.; BASTIEN, M. **Evidian Secure Access Manager Standard Edition**. [S.l.: s.n.], 2003. Evidian. dominique.castan@evidian.com; michel.bastien@evidian.com.

CERN / IT / PDP group. **SUE: standard unix environment at cern**. http://proj-sue.web.cern.ch/proj-sue/. Último acesso realizado em dezembro de 2004.

CERT/CC. **CERT Advisory CA-2002-03 Multiple Vulnerabilities in Many Implementations of the Simple Network Management Protocol (SNMP)**. http://www.cert.org/advisories/CA-2002-03.html. Último acesso realizado em dezembro de 2004.

CHRISTIANSEN, T.; TORKINGTON, N. **Perl Cookbook**. 1.ed. [S.l.]: O'Reilly & Associates, 1998.

CHUN, W. J. **Core Python Programming**. 1.ed. [S.l.]: Prentice Hall, 2000. 816p.

CONS, L.; POZNANSKI, P. Pan: A high-level configuration language. In: SYSTEMS ADMINISTRATION CONFERENCE (LISA-02), 16., 2002, Berkeley, CA. **Proceedings...** USENIX Association, 2002. p.83–98.

COOPER, M. **Advanced Bash-Scripting Guide**. An in-depth exploration of the art of shell scripting. thegrendel@theriver.com http://www.tldp.org/LDP/abs/html/. Último acesso realizado em agosto de 2004.

DARWIN, I. **Java Cookbook**. 1.ed. [S.l.]: O'Reilly, 2001.

DEITEL, H.; DEITEL, P.; SANTRY, S. **Advanced Java 2 Platform - How to Program**. [S.l.]: Prentice Hall, 2001.

Dell Inc. **Using NNM SE: hp openview network node manager special edition 1.0 with dell h ip 3.0**. http://support.dell.com/support/edocs/software/smnnmse/nnmse10/usingnnm.htm. Último acesso realizado em dezembro de 2004.

DEPPING, W. **Luma - LDAP browser, utility and more.** http://luma.sourceforge.net/about.html. Último acesso realizado em dezembro de 2004.

DERDELINCKX, K. **GQ is a GTK-based LDAP client**. http://biot.com/gq/.Últimoacessorealizadoemmarçode2004.

DIXON, W.; KIEHL, T.; SMITH, B.; CALLAHAN, M. **An Analysis of LDAP Performance Characteristics**. [S.l.: s.n.], 2002. http://www.crd.ge.com/cooltechnologies/pdf/2002grc154.pdf. Baixado em novembro de 2004.

DUNNAVANT, C. **Alchemist**. http://ictlab.tyict.vtc.edu.hk/ftp/rh-enterprise-SRPMS-updated-SPECS/al%chemist.spec, http://updates.cpubuilders.com/packages/index.php?mode=view\&name=alche%mist\&page=details\&pn=1. Último acesso realizado em dezembro de 2004.

DURáN, F. E. S. D. **CheckInstall**. http://asic-linux.com.mx/~izto/checkinstall/.

ENNS, R. **NETCONF Configuration Protocol**. [S.l.: s.n.], 2004. Internet Draft. http://www.ietf.org/internet-drafts/draft-ietf-netconf-prot-04.txt. Último acesso realizado em dezembro de 2004.

FILHO, R. H. **SAGRES - Um Sistema Baseado em Conhecimento para Apoio à Gerência de Falhas em Redes de Computadores**. 1998. 166 páginasp. Dissertação de mestrado em Ciência da Computação — Universidade Federal do Ceará — UFC, Ceará. http://www.mcc.ufc.br/disser/RaimirHolanda.pdf. Baixado em novembro de 2004.

FLEMING, P.; MCDONALD, I. **Lightweight Directory Access Protocol (LDAP): schema for printer services**. [S.l.: s.n.], 2004. RFC 3712. Category: Informational. http://www.faqs.org/rfcs/rfc3712.html.

FOLEY, R. **Perl Debugger Pocket Reference**. [S.l.]: O'Reilly & Associates, 2003.

FORD, R.; ELLIOT, S. **Extending ManageWise for the Challenges of the Enterprise**. www.novell.com/documentation.

FRANCESCHI, A. S. M. de; ROISENBERG, M.; BARRETO, J. M. Desenvolvendo Agentes de Software para Gerência de Redes Utilizando Técnicas de Inteligência Artificial. In: II CONGRESSO BRASILEIRO DE COMPUTAçãO, 2002, Itajaí, SC. **Anais...** [S.l.: s.n.], 2002. http://www.inf.ufsc.br/~barreto/artigos/Franceschi02.pdf. Baixado em outubro de 2004.

FRANKLIN, S.; GRAESSER, A. Is it an Agent, or just a Program?: a taxonomy for autonomous agents. In: INTELLIGENT AGENTS III. AGENT THEORIES, ARCHITECTURES AND LANGUAGES (ATAL'96), 1996, Berlin, Germany. **Anais...** Springer-Verlag, 1996. v.1193.

FRYE, R.; LEVI, D.; ROUTHIER, S.; WIJNEN, B. **Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework**. [S.l.: s.n.], 2000. RFC 2576. Category: Standars Track. http://www.faqs.org/rfcs/rfc2576.html.

FUJISAKI, T.; HAMADA, M.; KAGEYAMA, K. A Scalable Fault-tolerant Network Management System Built using Distributed Object Technology. In: INTERNATIONAL ENTERPRISE DISTRIBUTED OBJECT COMPUTING CONFERENCE, 1., 1997. **Anais...** IEEE, 1997. p.140–148.

GATWOOD, D. **Darwin NetInfo HOWTO**. Apple Technical Publications. dgatwood@apple.com. http://www.opendarwin.org/en/articles/netinfo/. Último acesso realizado em dezembro de 2004.

GFI Software Ltd. **GFI Network Server Monitor Overview**. http://www.gfi.com/nsm/?adclickid=2292752. Último acesso realizado em dezembro de 2004.

GITE, V. G. **Linux Shell Scripting Tutorial v1.05r3 - A Beginner's handbook**. http://www.freeos.com/guides/lsst/. Último acesso realizado em novembro de 2004.

GNU Operating System - Free Software Foundation. **gdbm**. http://www.gnu.org/software/gdbm/gdbm.html. Último acesso realizado em dezembro de 2004.

GOOD, G. **The LDAP Data Interchange Format (LDIF) - Technical Specification**. [S.l.: s.n.], 2000. RFC 2849. Category: Standards Track. http://www.faqs.org/rfcs/rfc2849.html.

GREENBLATT, B.; RICHARD, P. **An LDAP Control and Schema for Holding Operation Signatures**. [S.l.: s.n.], 1999. RFC 2649. Category: Experimental. http://www.faqs.org/rfcs/rfc2649.html.

Grupo de Gerência de Redes - Laboratório Metropoa / PUCRS. **Projeto FreeNMS - Requisitos e Especificações do Sistema**. [S.l.: s.n.], 2004. http://www.freenms.org/. Último acesso realizado em março de 2004.

GUELICH, S.; GUNDAVARAM, S.; BIRZNIEKS, G. **CGI Programming with Perl**. 2.ed. [S.l.]: O'Reilly & Associates, 2000.

GULBRANDSEN, A.; VIXIE, P.; ESIBOV, L. **A DNS RR for specifying the location of services (DNS SRV)**. [S.l.: s.n.], 2000. RFC 2782. Category: Standards Track. http://www.faqs.org/rfcs/rfc2782.html.

GÉLINA, J. **Linuxconf - Linux Administration Made Easy!** http://www.solucorp.qc.ca/linuxconf/. Último acesso realizado em março de 2003.

Hewlett-Packard Development Company, L.P. **HP OpenView Management Software**. http://openview.hp.com/downloads/index.html. Último acesso realizado em dezembro de 2004.

HILLEGASS, A. **Cocoa Programming for Mac OS X**. [S.l.]: Pearson Education, Inc., 2002.

HIROYASU, T.; MIKI, M.; KODAMA, K.; UEKAWA, J.; DONGARRA, J. A Simple Installation and Administration Tool for the Large-scaled PC Cluster System. In: CLUSTERWORLD CONFERENCE AND EXPO, SAN JOSE, CALIFORNIA, JUNE 24–26, 2003, 2003. **Anais...** [S.l.: s.n.], 2003.

HOCHSTETLER, S.; GLASMACHER, P.; MOELLER, M.; JEWAN, P.; PATHRE, M.; POLACHECK, J.; SAMPATH, V.; ZILLER, H. **Tivoli NetView 6.01 and Friends**. 1.ed. [S.l.]: IBM RedBooks. IBM International Technical Support Organization, 2000. 600p.

HODGES, J.; MORGAN, R. **Lightweight Directory Access Protocol (v3): technical specification**. [S.l.: s.n.], 2002. RFC 3377. Category: Standars Track. http://www.faqs.org/rfcs/rfc3377.html.

HODGES, J.; MORGAN, R.; WAHL, M. **Lightweight Directory Access Protocol (v3): extension for transport layer security**. [S.l.: s.n.], 2000. RFC 2830. Category: Standars Track. http://www.faqs.org/rfcs/rfc2830.html.

HOEPERS, C. Especialista defende maior capacitação dos profissionais. **Computação Brasil**, [S.l.], p.8–9, Dec 2004.

HOLGATE, M.; PARTAIN, W. The Arusha Project: a framework for collaborative unix system administration. In: LISA 2001 15TH SYSTEMS ADMINISTRATION CONFERECE, 2001. **Anais...** USENIX Association, 2001.

HOWARD, L. **An Approach for Using LDAP as a Network Information Service**. [S.l.: s.n.], 1998. RFC 2307. Category: Experimental. http://www.ietf.org/rfc/rfc2307.txt.

HOWES, T. A. **Understanding and Deploying LDAP Directory Services**. 1.ed. [S.l.]: Addison-Wesley Professional, 1999. 850p.

HOWES, T. A.; SMITH, M. C. **A Scalable, Deployable, Directory Service Framework for the Internet**. [S.l.]: Application Technology, Internet Society, 1995. http://www.isoc.org/HMP/PAPER/173/html/paper.html. Último acesso realizado em novembro de 2004.

HOWES, T. **The String Representation of LDAP Search Filters**. [S.l.: s.n.], 1997. RFC 2254. Category: Standards Track. http://www.faqs.org/rfcs/rfc2254.html.

HOWES, T.; SMITH, M. **The LDAP URL Format**. [S.l.: s.n.], 1997. RFC 2255. Category: Standars Track. http://www.faqs.org/rfcs/rfc2255.html.

IBM alphaWorks. **XML Data Mediator**. http://www.alphaworks.ibm.com/tech/XI?open\&l=NT101002,t=awfl. Último acesso realizado em novembro de 2003.

IBM Linux Technology Center. **Publications (By Project) — OpenLDAP**. http://www-124.ibm.com/linux/pubs/?project_id=61. Último acesso realizado em dezembro de 2004.

Ipswitch, Inc. **WhatsUp Professional**. http://www.ipswitch.com/Products/WhatsUp/professional/index.html. Último acesso realizado em dezembro de 2004.

ISQUIERDO, G. S. **Integração do Serviço de Diretório LDAP com o Serviço de Nomes CORBA**. 2001. 75 páginasp. Dissertação de mestrado em Ciência da Computação — Instituto de Matemática e Estatística da Universidade de São Paulo (IME/USP), São Paulo, SP.

JAVA, C. C. **An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program**. 2000.

JIANG, G. Multiple vulnerabilities in SNMP. **IEEE Computer — Security and Privacy**, [S.l.], v.35, n.4, p.2–4, Apr. 2002.

John Maler Thomson. **NetCentral Application Remotely Monitors Cable Equipment and Modems Through Customizable User Interface**. `http://www.thomson.net/EN/Home/Press/PressReleases/CorporatePress/PREN0%30807.htm`. Último acesso realizado em dezembro de 2004.

JOHNER, H.; BROWN, L.; HINNER, F.-S.; ; REIS, W.; WESTMAN, J. **Understanding LDAP**. [S.l.]: IBM RedBooks, International Technical Support Organization, 1998.

JOHNER, H.; MELOT, M.; STRANDEN, H.; WIDHIASTA, P. **LDAP Implementation Cookbook**. 1.ed. [S.l.]: IBM International Technical Support Organization, 1999.

JR, E. P. D.; MANSFIELD, G.; NANYA, T.; NOGUCHI, S. Improving the Dependability of Network Management Systems. **International Journal of Network Management**, [S.l.], v.8, p.244–253, 1998.

KERNIGHAN, B. W.; RITCHIE, D. M. **The C (ANSI C) Programming Language**. 2.ed. [S.l.]: Prentice Hall, 1988.

KILLE, S. **Representing the O/R Address hierarchy in the X.500 Directory Information Tree**. [S.l.: s.n.], 1995. RFC 1836. Category: Experimental. `http://www.faqs.org/rfcs/rfc1836.html`.

KILLE, S. **MHS use of the X.500 Directory to support MHS Routing**. [S.l.: s.n.], 1995. RFC 1801. Category: Experimental. `http://www.faqs.org/rfcs/rfc1801.html`.

KILLE, S. **Use of an X.500/LDAP directory to support MIXER address mapping**. [S.l.: s.n.], 1998. RFC 2164. Category: Standards Track. `http://www.faqs.org/rfcs/rfc2164.html`.

KIM, D.-H.; CHO, Y.-Z. An Efficient Integrated Network Management for Heterogeneous LANs and WANs. In: FOURTH INTERNATIONAL CONFERENCE ON HIGH-PERFORMANCE COMPUTING IN THE ASIA-PACIFIC REGION, 2000. **Anais...** [S.l.: s.n.], 2000.

KNOERNSCHILD, K. **Java Design: objects, uml, and process**. [S.l.]: Addison Wesley, 2001. 304p.

KONA, M. K.; XU, C.-Z. A Framework for Network Management using Mobile Agents. In: INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM (IPDPS'02), 2002. **Anais...** IEEE, 2002.

KOUTSONIKOLA, V.; VAKALI, A. LDAP: framework, practices, and trends. **IEEE Internet Computing**, [S.l.], v.8, n.5, p.66–72, 2004.

KUMAR, A. The OpenLDAP Proxy Cache. In: OPENLDAP DEVELOPERS' DAY, 2004., 2004. **Proceedings...** [S.l.: s.n.], 2004. `http://www.openldap.org/pub/kapurva/proxycaching.pdf`. Baixado em janeiro de 2004.

LAIRD, C.; SORAIZ, K. LDAP Comes of Age. **WebServer Magazine**, [S.l.], May 1999.

LANGE, T. **FAI Guide (Fully Automatic Installation)**. [S.l.]: Informatik und Wirtschaftsinformatik, Universität zu Köln, 2004. `lange@informatik.uni-koeln.de`.

LEE, B.; CHOI, T. **CLI-based Mediation Mechanism using XML for Configuration Management (draft-lee-xmlconf-xcli-00.txt)**. `http://www.ietf.org/internet-drafts/draft-lee-xmlconf-xcli-00.txt`. Último acesso realizado em agosto de 2003.

LEE, K.-H. Design and Implementation of an Integrated Network Management Agent. In: ICICS 2001, 2001. **Anais...** [S.l.: s.n.], 2001.

LEINWAND, A.; FANG, K. **Network Management: a practical perspective**. [S.l.]: Addison Wesley Professional, 1995. 352p.

LEMBURG, M.-A. **How Python is developed**. Slides of EuroPython Conference 2004, Göteborg, Sweden.

LERDORF, R.; TATROE, K. **Programming PHP**. [S.l.]: O'Reilly, 2002. 524p.

LIGNERIS, B. des; SCOTT, S.; NAUGHTON, T.; GORSUCH, N. Open Source Cluster Application Resources (OSCAR): design, implementation and interest for the [computer] scientific community. In: ANNUAL INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE COMPUTING SYSTEMS AND APPLICATIONS, 17., 2003. **Anais...** [S.l.: s.n.], 2003.

LINDEN, P. van der. **Expert C Programming: deep c secrets**. [S.l.]: Prentice Hall, 1994. 384p.

LONG, J. A Solution to the Problem of Configuration in Linux. **Freshmeat.net**, [S.l.], sep 2002. `http://freshmeat.net/articles/view/565/`. Último acesso realizado em dezembro de 2004.

LONG, J.; YACKOSKI, J. **Configuration 4 GNU (CFG)**. http://config4gnu.sourceforge.net/. Último acesso realizado em janeiro de 2005.

MARTIN, R. **Are scripting languages the wave of the future?** http://www.itworld.com/AppDev/1262/itw-0314-rcmappdevint/. Último acesso realizado em dezembro de 2004.

MCKECHNIE, M.; NAKAJIMA, K.; UELPENICH, S. **NetView Distribution Manager/6000 Release 1.2 Agents and Advanced Scenarios**. 1.ed. [S.l.]: IBM RedBooks. IBM International Technical Support Organization, 1995. 387p.

MOFFETT, J. D.; SLOMAN, M. S. Policy Hierarchies for Distributed Systems Management. **IEEE JSAC Special Issue on Network Management**, [S.l.], v.11, n.9, p.1404–1414, Dec. 1993.

MOFFETT, J. D.; SLOMAN, M. S.; TWIDLE, K. P. Specifying Discretionary Access Control Policy for Distributed Systems. **IEEE Computer Communications**, [S.l.], v.13, n.9, p.571 – 580, Nov. 1990.

MULLER, N. J. Improving Network Operations With Intelligent Agents. **International Journal of Network Management**, [S.l.], v.7, p.116 – 126, 1997.

MURDOCH, J. L. **Distributed, Agent-Based, Network Fault Diagnosis System**. 2003. 68p. Thesis contributing to hand-in requirements of Napier University in the BEng with Honours Computer Networks and Distributed Systems Degree — School of Computing, Napier University, Edinburgh, UK.

MYERS, J. **Simple Authentication and Security Layer (SASL)**. [S.l.: s.n.], 1997. RFC 2222. Category: Standars Track. http://www.faqs.org/rfcs/rfc2222.html.

n software inc. **IP*Works! V6 Delphi Edition**. http://www.nsoftware.com/products/showprod.aspx?part=IPD6-A. Último acesso realizado em novembro de 2004.

NAUGHTON, P.; SCHILDT, H. **Java 2 The Complete Reference**. 4.ed. [S.l.]: Osborne/McGraw-Hill, 1999.

Novell, Inc. **Novell Developer Kit**. http://developer.novell.com/ndk/. Último acesso realizado em novembro de 2004.

Novell, Inc. **LDAP Libraries for C - NetWare and Windows**. http://developer.novell.com/ndk/cldap.htm. Último acesso realizado em novembro de 2004.

Novell, Inc. **NetWare LANalyzer Agent Installation and Administration Guide**. ManageWise 2.7. Management Software. www.novell.com/documentation.

Novell, Inc. **Setup Guide**. ManageWise 2.7. Management Software. www.novell.com/documentation.

Novell, Inc. **Network Management Guide**. ManageWise 2.7. Management Software. www.novell.com/documentation.

Novell, Inc. **Desktop Management Guide**. ManageWise 2.7. Management Software. www.novell.com/documentation.

Novell, Inc. **LDAP Configuration Tool**. http://forge.novell.com/modules/xfmod/project/?ldapcfgtool. Último acesso realizado em dezembro de 2004.

NUNES, R. C. **Adaptação Dinâmica do *timeout* de Detectores de Defeitos através do Uso de Séries Temporais**. 2003. 164p. Tese apresentada como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação — Programa de Pós-Graduação em Computação, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brasil.

NUNES, R. C.; BORTOLAS, A.; SILVA, R. L.; TURCHETTI, R. **AFDService - Adaptive Failure Detector Service**. http://www.inf.ufsm.br/~ceretta/projetos.html. Último acesso realizado em janeiro de 2005.

OctetString, Inc. **JDBC-LDAP Bridge Driver**. http://www.octetstring.com/products/BridgeDriver.php. Último acesso realizado em dezembro de 2004.

OpenLDAP Foundation. **OpenLDAP**. http://www.openldap.org/. Último acesso realizado em janeiro de 2005.

OpenLDAP Foundation. **OpenLDAP Technical Support**. http://www.openldap.org/support/. Último acesso realizado em dezembro de 2004.

O'Reilly & Associates. **Python Success Stories**. [S.l.]: O'Reilly & Associates, 2003. http://python.oreilly.com/news/python_success_stories.pdf. Baixado em novembro de 2004.

OSTERLUND, R. PIKT: problem informant/killer tool. In: SYSTEMS ADMINISTRATION CONFERENCE (LISA2000), 14., 2000. **Anais...** USENIX Association, 2000.

OUSTERHOUT, J. K. Scripting: higher-level programming for the 21st century. **IEEE Computer Magazine**, [S.l.], v.31, n.3, p.23–30, 1998.

PADL Software Pty Ltd. **MigrationTools**. http://www.padl.com/OSS/MigrationTools.html. Último acesso realizado em outubro de 2003.

Paessler GmbH. **IPCheck Server Monitor**. http://www.paessler.com/ipcheck/?banner=GoogleIPCContent. Último acesso realizado em dezembro de 2004.

PAPADOPOULOS, P. M.; KATZ, M. J.; BRUNO, B. NPACI Rocks: tools and techniques for easily deploying manageable Linux clusters. **Concurrency and Computation: Practice and Experience**, [S.l.], v.15, n.7–8, p.707–725, June/July 2003.

PeopleSoft, Inc. **PeopleSoft - The world's most flexible, adaptable software**. `http://www.peoplesoft.com/corp/en/public_index.jsp`.

phpLDAPadmin Supporters. **Web-based LDAP browser to manage your LDAP server**. `http://phpldapadmin.sourceforge.net/`. Último acesso realizado em maio de 2004.

PILS, C. Agent based profile management. In: WSEAS SIM-SSIP-MIV-RODLICS 2001, MALTA, 2001. **Anais...** WSEAS, 2001.

PUJOLLE, G.; DUARTE, M. B.; RUBINSTEIN, M. G.; CARLOS, O. **Using Mobile Agent Strategies for Reducing the Response Time in Network Management**. [S.l.: s.n.], 2001.

PUJOLLE, G.; RUBINSTEIN, M. G.; DUARTE, D. **Evaluating the Network Performance Management Based on Mobile Agents**. [S.l.: s.n.], 2001.

PULIAFITO, A.; TOMARCHIO, O. Using Mobile Agents to implement flexible Network Management strategies. **Computer Communication Journal**, [S.l.], v.23, n.8, p.708–719, apr 2002.

Quest Software. **Big Brother**. `http://www.bb4.org/`, `http://www.quest.com/bigbrother/`. Último acesso realizado em dezembro de 2004.

REINKING, N. **Kantan LDAP Search Engine**. `http://www.twoevils.org/html/files.html`. Último acesso realizado em dezembro de 2004.

RICHICHI, M. How to Use Perl, Python and PHP to Access NDS eDirectory 8.5 via LDAP? **Novell Appnotes**, Assistant Director of Academic Technology, Drew University, 2001. `http://developer.novell.com/research/appnotes/2001/august/05/a010805.pd%f`. Baixado em novembro de 2004.

RUBINSTEIN, M. G.; DUARTE, O. C. M. B.; PUJOLLE, G. Evaluating the Performance of a Network Management Application Based on Mobile Agents. In: IFIP NETWORKING, 2002, Pisa, Italy. **Anais...** [S.l.: s.n.], 2002. p.515–526.

SCHAUMANN, J. **The snmpmon Handbook**. `http://www.netmeister.org/apps/snmpmon/`. Último acesso realizado em dezembro de 2004.

SCHMIDT, S. **Web Based System Administration with Webmin**. `http://www.heise.de/ct/english/98/14/068/`. Último acesso realizado em dezembro de 2004.

SCHREIBER, W. Building eDirectory-Enabled Applications using Delphi: low-level. **Novell Appnotes**, [S.l.], 2002. wschreiber@novel.com, http://developer.novell.com/research/appnotes/2002/february/03/a020203.%pdf. Baixado em novembro de 2004.

SCHWARTZ, R.; PHOENIX, T. **Learning Perl Objects, References & Modules** . 1.ed. [S.l.]: O'Reilly & Associates, 2003.

SEGURA, S. **System Management Interface Tool (SMIT)**. [S.l.: s.n.], 2000. http://www-1.ibm.com/servers/aix/products/aixos/whitepapers/smit.pdf. Baixado em agosto de 2003.

SESSINK, O. **Directory Assistant**. http://olivier.sessink.nl/directoryassistant/. Último acesso realizado em dezembro de 2004.

SHYNE, S.; MARKLE, H. Managing Heterogeneous Networks Across Security/Coalition Domains. In: MILCOM 2000. 21ST CENTURY MILITARY COMMUNICATIONS CONFERECE, 2000. **Anais...** [S.l.: s.n.], 2000. v.1, p.430–434.

SKYRIX Software AG. **SKYRiX Object Publishing Environment**. http://sope.opengroupware.org/index.html. http://sope.opengroupware.org/en/sope_ldap/ (SOPE Libraries for LDAP Connection). Último acesso realizado em dezembro de 2004.

SleepyCat Software, Inc. **Berkeley DB**. http://www.sleepycat.com/. Último acesso realizado em dezembro de 2004.

SMITH, M. **The Mozilla LDAP C SDK**. http://www.mozilla.org/directory/csdk.html. Último acesso realizado em dezembro de 2004.

SMITH, M.; HOWES, T.; HERRON, A.; WAHL, M.; ANANTHA, A. **The C LDAP Application Program Interface**. Internet Draft. http://www.mozilla.org/directory/ietf-docs/draft-ietf-ldapext-ldap-c-ap%i-05.txt.

SoftIntegration, Inc. **Ch LDAP Package for OpenLDAP**. http://chldap.sourceforge.net/. Último acesso realizado em dezembro de 2004.

SoftIntegration, Inc. **C for system programming; C++ for large-scale projects; C/C++ interpreter — Ch for script computing**. http://www.softintegration.com/. Último acesso realizado em dezembro de 2004.

SOLBRIG, H. R.; CHUTE, C. G. Terminology Access Methods Leveranging LDAP Resources. In: MEDINFO 2004, 2004. **Anais...** IOS Press: Amsterdam, 2004.

SPAINHOUR, S.; SIEVER, E.; PATWARDHAN, N. **Perl in a Nutshell**. 2.ed. [S.l.]: O'Reilly & Associates, 2002.

STAMME, H.; GAO, L. X.; GUIMARAES, M.; KENNEDY, C.; WYER, J. **Automation Using Tivoli NetView OS/390 V1R3 and System Automation OS/390 V1R3**. 1.ed. [S.l.]: IBM Red-Books. IBM International Technical Support Organization, 2000. 600p.

STEIN, L. D. **Network Programming with Perl**. [S.l.]: Addison Wesley Professional, 2000. 784p.

STOKES, E.; BYRNE, D.; BLAKLEY, B.; BEHERA, P. **Access Control Requirements for LDAP**. [S.l.: s.n.], 2000. RFC 2820. Category: Standars Track. `http://www.faqs.org/rfcs/rfc2820.html`.

STOKES, E.; WEISER, R.; MOATS, R.; HUBER, R. **Lightweight Directory Access Protocol (version 3) Replication Requirements**. [S.l.: s.n.], 2002. RFC 3384. Category: Informational. `http://www.faqs.org/rfcs/rfc3384.html`.

STRöDER, M. **web2ldap - WWW gateway to LDAP server**. `http://www.web2ldap.de/`. Último acesso realizado em dezembro de 2004.

Sun Microsystems. **Sun Management Center 3.5**. `http://www.sun.com/software/solaris/sunmanagementcenter/sunmc35u1.html`. Último acesso realizado em novembro de 2004.

Sun Product Documentation. **Sun Java Desktop System Configuration Manager, Release 1 — Using OpenLDAP and Active Directory with the Configuration Manager**. `http://docs.sun.com/app/docs/doc/819-0635/6n322p07h?a=view`. Último acesso realizado em dezembro de 2004.

SunSoft. **SunNet Manager 2.2.3 Reference Manual**. [S.l.]: Sun Microsystems, Inc., 1995. 382p. `http://docs.sun.com`.

SunSoft. **SunNet Manager 2.2.3 User's Guide**. [S.l.]: Sun Microsystems, Inc., 1995. 270p. `http://docs.sun.com`.

SVENTEK, J. **Configuration, Monitoring and Management of Huge-scale Applications with Varying Number of Applications Components**. `http://www.dcs.ed.ac.uk/home/dcspaul/wshop/HugeScale.pdf`. Baixado em novembro de 2004.

The Mozilla Organization. **PerLDAP Tools**. `http://www.mozilla.org/directory/tools/ldaptools.html`. Último acesso realizado em dezembro de 2004.

The OpenLDAP Project. **OpenLDAP 2.2 Administrator's Guide**. [S.l.: s.n.], 2004. `http://www.openldap.org/doc/admin22/`. Último acesso realizado em novembro de 2004.

The Perl Foundation. **The Perl Directory at Perl.org**. http://www.perl.org/.

The PHP Group. **LDAP Functions**. http://br2.php.net/ldap.

THONTON, E. J.; MUNDY, D.; CHADWICK, D. W. A Comparative Performance Analysis of 7 Lightweight Directory Access Protocol Directories. In: TERENA NETWORKING CONFERENCE, 2003, Zagreb, Croatia. **Anais...** [S.l.: s.n.], 2003. http://www.terena.nl/conferences/tnc2003/programme/papers/p1d1.pdf. Baixado em novembro de 2004.

TISDALL, J. **Mastering Perl for Bioinformatics**. [S.l.]: O'Reilly & Associates, 2003.

TSAI, E. **Netscape Directory SDK for Java: source code release**. http://www.mozilla.org/directory/javasdk.html. Último acesso realizado em dezembro de 2004.

UEKAWA, J. **Update-cluster usage guide**. http://www.netfort.gr.jp/~dancer/software/downloads/update-cluster-0.41%.0.tar.gz. Último acesso realizado em novembro de 2003.

University of Michigan. **The SLAPD and SLURPD Administrator's Guide**. [S.l.: s.n.], 1996. http://www.umich.edu/~dirsvcs/ldap/doc/guides/slapd/toc.html. Último acesso realizado em janeiro de 2005.

VALDÉS, J. Automating Software and Patch Installation With Cfengine. In: CONFIGURATION WORKSHOP - LISA2002, 2002. **Anais...** [S.l.: s.n.], 2002.

VIRTANEN, T. **Ldaptor home page**. http://tv.debian.net/software/ldaptor/. Último acesso realizado em dezembro de 2004.

VIRTANEN, T. **Ldaptor**. http://tv.debian.net/software/ldaptor/. Último acesso realizado em dezembro de 2004.

WAHL, M. **A Summary of the X.500(96) User Schema for use with LDAPv3**. [S.l.: s.n.], 1997. RFC 2256. Category: Standards Track. http://www.faqs.org/rfcs/rfc2256.html.

WAHL, M. **MIME Directory Profile for LDAP Schema**. [S.l.: s.n.], 2000. RFC 2927. Category: Information. http://www.faqs.org/rfcs/rfc2927.html.

WAHL, M.; ALVESTRAND, H.; HODGES, J.; MORGAN, R. **Authentication Methods for LDAP**. [S.l.: s.n.], 2000. RFC 2829. Category: Standars Track. http://www.faqs.org/rfcs/rfc2829.html.

WAHL, M.; COULBECK, A.; HOWES, T.; KILLE, S. **Lightweight Directory Access Protocol (v3): attribute syntax definitions**. [S.l.: s.n.], 1997. RFC 2252. Category: Standards Track. http://www.faqs.org/rfcs/rfc2252.html.

WAHL, M.; HOWES, T.; KILLE, S. **Lightweight Directory Access Protocol (v3)**. [S.l.: s.n.], 1997. RFC 2251. Category: Standars Track. http://www.faqs.org/rfcs/rfc2251.html.

WANG, X.; SCHULZRINNE, H.; KANDLUR, D.; VERMA, D. Measurement and analysis of LDAP performance. In: SIGMETRICS '00: PROCEEDINGS OF THE 2000 ACM SIGMETRICS INTERNATIONAL CONFERENCE ON MEASUREMENT AND MODELING OF COMPUTER SYSTEMS, 2000. **Anais...** ACM Press, 2000. p.156–165.

WARRENE, B. **Use Webmin for Linux Administration, Parts 1 and 2**. http://www.sitepoint.com/article/webmin-linux-administration-1. Último acesso realizado em dezembro de 2004.

Web Communications, ETT/EC-EX. **CERN - The world's largest particle physics laboratory**. http://www.cern.org. Último acesso realizado em dezembro de 2004.

WEBER, S. **Securing LDAP Through TLS/SSL — A Cookbook**. [S.l.: s.n.], 2002. http://www.sun.com/blueprints.

WILLIAMS, A. T. LDAP and OpenLDAP. In: OPENLDAP DEVELOPERS' DAY, 2003, Vienna (Wien), Austria (Österreich). **Anais...** [S.l.: s.n.], 2003. Presentation Transparencies.

YAACOVI, Y.; WAHL, M.; GENOVESE, T. **Lightweight Directory Access Protocol (v3): extensions for dynamic directory services**. [S.l.: s.n.], 1999. RFC 2589. Category: Standars Track. http://www.faqs.org/rfcs/rfc2589.html.

YEMINI, Y. The OSI Network Management Model. **IEEE Communications Magazine**, [S.l.], v.31, n.5, p.20–29, May 1993.

YEONG, W.; HOWES, T.; KILLE, S. **Lightweight Directory Access Protocol**. [S.l.: s.n.], 1995. RFC 1777. Category: Standards Track. http://www.faqs.org/rfcs/rfc1777.html.

ZANDSTRA, M. **Teach Yourself PHP in 24 Hours**. 3.ed. [S.l.]: Sams Publishing, 2003. 500p.

ZEILENGA, K. **OpenLDAP Root Service An experimental LDAP referral service**. [S.l.: s.n.], 2001. RFC 3088. Category: Experimental. http://www.faqs.org/rfcs/rfc3088.html.

ZEILENGA, K. **LDAP Password Modify Extended Operation**. [S.l.: s.n.], 2001. RFC 3062. Category: Standars Track. http://www.faqs.org/rfcs/rfc3062.html.

ZEILENGA, K. **LDAP Authentication Password Schema**. [S.l.: s.n.], 2001. RFC 3112. Category: Informational. http://www.faqs.org/rfcs/rfc3112.html.

Zetwork GmbH. **LDAPUserFolder**. http://www.dataflake.org/software/ldapuserfolder, http://www.zetwork.com/en/index.html. Último acesso realizado em dezembro de 2004.

Zope Community. **The Web Site for the Zope Community**. http://www.zope.org/. Último acesso realizado em dezembro de 2004.

APÊNDICE A FERRAMENTAS DE GERENCIAMENTO E MONITORAMENTO

Este apêndice apresenta informações complementares ao capítulo 2. São comentadas em maiores detalhes as características desejáveis em sistemas de gerenciamento integrado de redes de computadores e os trabalhos relacionados.

A próxima seção apresenta brevemente algumas das características desejáveis em sistemas de gerenciamento geral de redes de computadores. Atualmente são poucas as ferramentas que englobam grande parte dos recursos discriminados na seção seguinte. Isso demonstra que a área de gerenciamento de redes está em constante evolução e desenvolvimento.

As demais seções apresentam as quatro classificações básicas (ver capítulo 2) e um descrição das ferramentas situadas em cada uma destas categorias. Na seção A.5 são também apresentadas alguns sistemas que não se enquadram nesses quatro conjuntos de ferramentas.

A.1 Protocolos

As seções que seguem abordam alguns protocolos conhecidos e utilizados por administradores de domínios.

A.1.1 SNMP

O SNMP (Simple Network Management Protocol) (CASE et al., 1990; FILHO, 1998) é um protocolo de padrão gerenciamento de redes e largamente aceito pelo mercado e pelo meio acadêmico. O objetivo deste protocolo é fornecer um conjunto de definições básicas e simples para o gerenciamento e monitoramento de dispositivos de rede.

O modelo de gerenciamento SNMP (AZAMBUJA, 2001) é baseado na arquitetura cliente-agente, onde o agente é o elemento de gerenciamento disponível em cada dispositivo de rede gerenciável. O cliente é uma estação de monitoramento que realiza consultas e envia requisições e dados ao agente. Com essa arquitetura é possível realizar operações padrão de monitoramento e configuração dos componentes de uma rede.

Várias soluções paralelas ao SNMP estão surgindo, devido a este protocolo ser simples demais para o gerenciamento completo de sistemas e dispositivos de rede. Além disso, a escala-

bilidade deste protocolo é baixa devido a sua extrema simplicidade (ADHICANDRA; PATTINSON; SHAGHOUCI, 2003).

Outros problemas inerentes ao SNMP são:

- operações restritas de comunicação e gerenciamento do protocolo;
- MIBs (Management Information Base) são estáticas e armazenadas localmente em cada dispositivo de rede. Elas também são pouco flexíveis para a representação de tipos de dados variáveis, a critério do administrador do sistema. Além disso, a alteração de uma MIB implica a recompilação da mesma, o que pode ser uma tarefa pouco desejável em um ambiente dinâmico e complexo de gerenciamento de sistemas.
- o protocolo possuí vários problemas de segurança no protocolo (JIANG, 2002; CERT/CC, 2004);
- o SNMP é pouco adaptável para o gerenciamento de ambientes diversificados e dinâmicos, onde os administradores de sistemas desejam (ou precisam) criar soluções específicas e facilmente extensíveis. Nestes aspectos, faltam funcionalidades no protocolo (YEMINI, 1993). Esse foi um dos motivos que levou à criação do NetConf (ver seção A.1.2).

Outros tipos de falhas possíveis no protocolo são quanto a problemas de comunicação (BURGESS; CANRIGHT, 2003). Como o SNMP usa UDP para a comunicação, caso algum esquema de configuração dependa da disponibilidade de um recurso ou componente em um determinado momento as chances de as operações serem bem sucedidas são limitadas (BURGESS; CANRIGHT, 2003).

A.1.2 NetConf

O NetConf (Active Working Group, 2003; ADWANKAR, 2004; ENNS, 2004) é um novo protocolo de gerenciamento de dispositivos de rede. Sua formulação iniciou em meados de 2003 e continua sendo desenvolvido. Provavelmente será uma boa opção em relação a protocolos mais simples, como o SNMP.

Entre os objetivos deste novo protocolo estão: (Active Working Group, 2003)

- prover mecanismos de acesso à informação que sejam capazes de diferenciar entre dados de configuração e dados que não são de configuração;
- ser extensível o suficiente para que vendedores possam prover acesso a todos os dados de configuração de um dispositivo através de um único protocolo;
- utilizar representação textual de dados, que pode ser facilmente manipulada através do uso de ferramentas não especializadas de manipulação de texto;
- suportar integração de métodos existentes de autenticação de usuários;

- suportar integração com sistemas de base de dados de configuração existentes;
- suportar transações de configuração em nível da Internet;
- suportar notificações assíncronas.

Em suma, NetConf é um protocolo em fase de estudo e projeto, cujo objetivo aparentemente é apenas configurar dispositivos de rede e não ser um sistema mais abrangente e flexível para o gerenciamento de serviços e sistemas de um modo geral.

A.2 Linguagens de Configuração

Esta seção apresenta alguns sistemas, com linguagens de descrição e configuração de dados próprias, de gerenciamento de redes de computadores.

A.2.1 PAN

Pan é apenas uma linguagem para expressar informação de configuração (CONS; POZNANSKI, 2002). Este sistema não pode ser considerado uma ferramenta de administração propriamente dita.

A linguagem própria do Pan é capaz de realizar uma verificação forte de consistência e integridade sobre os dados de configuração. Para tanto, possui um compilador próprio. Este é responsável por recriar os conjuntos de informações de baixo nível quando alguma relação de dependência em alto nível é modificada.

Resumidamente, esta é basicamente uma linguagem para a verificação e manutenção das configurações de sistemas e não um sistema para um gerenciamento abrangente de redes de computadores.

A.2.2 PIKT

PIKT (*Problem Informant/Killer Tool*) (OSTERLUND, 2000) é uma linguagem de configuração que acompanha um interpretador próprio. PIKT também é um *script* sofisticado e um pré-processador para arquivos de configuração de sistema para ser utilizado com a linguagem PIKT ou outra linguagem interpretada qualquer (OSTERLUND, 2000). PIKT é ainda um agendador central (estilo o *cron* de sistemas Unix) para *scripts*, com execução periódica e um instalador customizado.

O propósito principal do PIKT é monitorar sistemas, registrar falhas e fixar os problemas quando possível (OSTERLUND, 2000). Em uma configuração usual, funciona na forma mestre-escravo, ou seja, as máquinas escravas são monitoradas através de uma máquina mestra. Algumas das funcionalidades incluem a instalação de arquivos em clientes-alvos e operações como inicialização e finalização da execução de demônios de sistema.

Contudo, PIKT não é um sistema geral de configuração e não provê muitas das características esperadas em uma ferramenta de gerenciamento abrangente e adaptável.

A.2.3 Cfengine

Cfengine (BURGESS, 1995, 2001, 2004) é uma ferramenta que permite a automatização várias tarefas simples de administradores de sistemas. Esta ferramenta incorpora uma linguagem declarativa mais abstrata que linguagens como Perl e *Shell Scripts*. Esta ferramenta é boa para realizar tarefas administrativas comuns, como verificação de *links* e cópia de arquivos (BURGESS, 2001).

O maior objetivo do Cfengine é permitir a criação de um sistema único e centralizado de configuração, que irá definir como cada máquina da rede deve ser configurada. A configuração de cada computador será verificada em relação a descrição geral de configuração (BURGESS, 2001).

Gerenciar os arquivos de configuração do Cfengine pode tornar-se uma tarefa difícil e complexa. Ordenamento pode vir a ser um se o administrador da rede não for cuidadoso, por exemplo. Muitas vezes, pode ser difícil de realizar mudanças simples manualmente (VALDÉS, 2002). Neste caso, o administrador pode ser forçado a utilizar a linguagem declarativa e a seguir a ordem de precedência das diretivas de declaração do Cfengine.

Os dados da ferramenta não são reais abstrações da configuração de uma máquina. Mas sim, são instruções para os diferentes módulos como a configuração de interfaces de rede, monitoramento de permissões de arquivos, gerenciamento de *links* simbólicos, adição de linhas em arquivos de configuração de sistemas, entre outras funcionalidades. A modificação de um arquivo de configuração pode implicar instruções como `AppendIfNoSuchLine` ou `CommentLinesMatching` (CONS; POZNANSKI, 2002; BURGESS, 1995). Isso pode levar a construções complexas e difíceis de serem mantidas.

O objetivo deste sistema não é prover mecanismos para a geração e a atualização de sistemas. Aspectos como flexibilidade, integração, ferramentas de apoio e granularidade administrativa estão também entre os pontos fracos da ferramenta.

A.2.4 LCFG

O *framework* LCFG (ANDERSON, 1994; ANDERSON; SCOBIE, 2002; ANDERSON, 2001a) foi desenvolvido pelo departamento de ciência da computação da universidade de Edinburgh, com o objetivo de gerenciar algumas centenas de máquinas Unix. Inicialmente, o sistema era suportado em Solaris, posteriormente, foi migrado para GNU/Linux e é mantido somente neste último (ANDERSON; SCOBIE, 2002).

O LCFG suporta mais de 2000 parâmetros de configuração. Usava mapas NIS (ANDERSON, 1994) e usa HTTP (ANDERSON; GOLDSACK; PATERSON, 2003). Mesmo assim, aspectos de escalabilidade não são amplamente abordados.

A centralização das configurações é baseada em arquivos em um servidor central (ANDERSON; SCOBIE, 2002). No entanto, manter muitos arquivos, especialmente de diferentes formatos, é trabalhoso e pode resultar em falhas de configuração indesejáveis. Além disso, não há relação e dependência entre as diferentes informações distribuídas pelos arquivos de configura-

ção específicos a cada aplicativo gerenciado.

A.2.5 Arusha

O projeto Arusha (Arusha Project Team, 2003; HOLGATE; PARTAIN, 2001) provê um *framework* para a administração colaborativa de domínios com plataformas Unix diversificadas. A ferramenta provê uma linguagem simples, baseada em XML, para a descrição de configurações e ambientes, desde o gerenciamento de pacotes até o gerenciamento de documentação ou configuração de sistemas.

A administração colaborativa de sistemas do Arusha surgiu devido a questões como: muitos administradores de sistemas trabalham em isolamento e, por isso, estão a toda hora reinventando a roda. Mais ainda, as freqüentes soluções não-revistas e revisadas são de baixa qualidade. Muitas vezes, estas soluções surgiram em situações emergenciais e carecem de consistência.

A idéia principal do Arusha é contribuir no armazenamento, organização e manutenção das informações do domínio administrativo, provendo o necessário para estabelecer um gerenciamento colaborativo.

A pesar disso, a linguagem do Arusha sofre de validação e checagem fortes (CONS; POZNANSKI, 2002). Ela também mistura código e dados gerais de configuração (exemplo: código Perl ou Python pode ser inserido no XML, perto dos dados).

A.2.6 Alchemist

O Alchemist (DUNNAVANT, 2000) é uma arquitetura transparente de configuração. Ela provê configurações de múltiplas fontes em nível de dados, retardando a tradução para um formato nativo em último estágio. XML é utilizado para a codificação intermediária de dados e pode ser estendida arbitrariamente a vários cenários de configuração (DUNNAVANT, 2000).

Poucos detalhes e pouca documentação são fornecidos sobre esta arquitetura de configuração (ANDERSON, 2001b).

A.3 Ferramentas de Gerenciamento Remoto

Esta seção apresenta algumas ferramentas de gerenciamento remoto de sistemas de uma rede de computadores.

A.3.1 NetInfo

O NetInfo facilita o gerenciamento e a administração da informação utilizada por computadores Mac OS X (Apple Computer, Inc., 2001). Este sistema possibilita a centralização da informação acerca de usuários, impressoras, servidores e outros dispositivos de rede de forma que todos os computadores Mac OS X, ou apenas alguns deles, tenham acesso a esses dados. A centralização da informação com o NetInfo ajuda na configuração e no gerenciamento de

diretórios de usuários em múltiplos e integrados servidores Mac OS X.

A base de dados do NetInfo pode também ser utilizada para a autenticação e administração remota. No entanto, por razões de segurança isto não é recomendado. O uso de tecnologias como LDAP seria uma opção mais conveniente e segura (GATWOOD, 2004).

Tecnologias como o LDAP apresentam maiores características em relação a segurança, granularidade administrativa, mecanismos de autorização, escalabilidade e replicação nativa. Além disso, LDAP é uma tecnologia atualmente mais difundida e utilizada que o NetInfo. Isso pode ser visto através do número de organizações que estão dando suporte ao LDAP e do número de ferramentas disponíveis para a manipulação e uso de seus diretórios de dados (ver seção 4.2).

A.3.2 Config4gnu

O projeto configuração para GNU (Configuration 4 GNU) (LONG; YACKOSKI, 2002) tem por objetivo amenizar muitas das tarefas de gerenciamento de configurações de sistemas Linux/Unix. A idéia é prover conjuntos de ferramentas para usuários, administradores e desenvolvedores que sejam capazes de tornar tarefas de configuração e manutenção de sistemas mais eficientes e poderosas. Isso tudo sem sair da filosofia Unix de simplicidade e independência.

Os objetivos gerais do projeto são (LONG; YACKOSKI, 2002):

- prover interfaces múltiplas que permitam a usuários e administradores modificarem configurações;
- permitir que desenvolvedores criem facilmente interfaces de configuração para casos específicos do domínio administrativo;
- padronizar os diferentes tipos de arquivos de configuração de sistemas e prover conversores de formato para todos os estilos;
- incluir mecanismos que atendam às necessidades de diferentes sistemas operacionais e suas distribuições derivadas;
- manter os arquivos de configuração nativos (expressos em linguagem de alto nível, compreensível por humanos) no diretório `/etc` como cópia autoritativa da configuração do sistema;
- criar um sistema que é geral ao ponto de permitir níveis elevados de flexibilidade, mas também especializado o suficiente para que tarefas específicas de configuração possam ser realizadas de maneira prática e eficiente.

A intenção do Config4gnu não é ter uma única base de informações, mas sim, diversas formas de representar os dados de configuração. As configurações podem ser mantidas em arquivos de texto simples, em descrições XML e em formatos específicos de configuração de

sistemas. Neste caso, a intenção é manter os formatos de configuração nativos de serviços e aplicações em geral.

A não-integração e falta de uniformidade das informações pode prejudicar processos automáticos e dinâmicos de manutenção e atualização das configurações de sistemas. Outro aspecto desfavorável é a manutenção de vários formatos e tipos de descrições de configuração, pois isso pode acabar tornando complicada a tarefa de manter o próprio sistema.

A.3.3 ManageWise

ManageWise (FORD; ELLIOT, 1996; Novell, Inc., 1999a,b) é uma plataforma composta de vários componentes para o gerenciamento de redes de diferentes vendedores como se fossem um único sistema. É desenvolvido e comercializado pela Novell.

Entre as características da ManageWise podem ser citadas (Novell, Inc., 1999b):

- auto descobrimento de dispositivos de rede no ambientes IP, IPX ou misto (IP e IPX);
- monitoramento e gerenciamento de servidores, roteadores e HUBs;
- notificação através de alarmes sobre potenciais problemas na rede;
- fornecimento de uma visão gráfica da rede, possibilitando controlar e visualizar informações sobre dispositivos;
- gerenciamento restrito a servidores NetWare, através do SNMP (Novell, Inc., 1999c).

O ManageWise é suportado nas plataformas Windows e NetWare (Novell, Inc., 1999d). Gerenciamento remoto é possível através da Console ManageWise. No entanto, este processo é manual e pouco prático para a administração simultânea e integrada de um grande número de máquinas. Esta ferramenta não é um sistema geral de configuração e manutenção de sistemas.

A.3.4 SunNet Manager

SunNet Manager (SNM) (SunSoft, 1995a,b) é um conjunto de ferramentas e serviços que podem ser utilizados para a realização de tarefas fundamentais em uma rede. O SNM consiste em uma plataforma para o gerenciamento de grupos de redes distribuídas (SunSoft, 1995a). Possui mecanismos para estender a plataforma. Com isto ela torna-se hábil à continuidade de um gerenciamento efetivo mesmo com mudanças de requisitos e necessidades. A base do SNM está sobre o modelo gerente/agente descrito no *framework* de gerenciamento da OSI e sobre o protocolo TCP/IP nativo do Solaris. O gerente é ativado pelo usuário. O agente é um processo que acessa objetos gerenciáveis e coleta dados em benefício do gerente (SunSoft, 1995a).

O gerenciamento é baseado no SNMP, ou seja, restrito à modificação de atributos e comandos do SNMP. Mesmo assim, possui duas bases de dados: MDB (Management DataBase) e a base de dados em tempo de execução (SunSoft, 1995b). Esta última pode ser alterada dinamicamente, enquanto que a primeira pode ser editada manualmente e não é dinâmica. Além disso,

algumas operações somente podem ser feitas editando diretamente o arquivo da MDB. Entre elas:

- adição e modificação de tipos de elementos;
- especificação de novos ícones para máquinas para um tipo de elemento;
- adição ou modificação de ferramentas em um menu de um ícone de um elemento;

O SNM não é uma ferramenta geral de gerenciamento e manutenção de redes de computadores. Além disso, não é adaptável a contextos diversificados.

A.3.5 SUE

SUE significa *Standard Unix Environment* (CERN / IT / PDP group, 2000) é um conjunto de componentes, arquivos de configuração e programas utilitários que juntos formam um sistema Unix customizado e pronto para ser usado.

Um dos principais objetivos de SUE é ser apropriado para o maior número possível de computadores Unix, de uma determinada arquitetura, no CERN (Web Communications, ETT/EC-EX, 2004). É esperado que um conjunto comum de componentes de sistema instalados e configurações similares sejam o suficiente para simplificar instalações, reduzir as possibilidades de configurações errôneas e resolução de problemas conhecidos, diminuir a necessidade de habilidades específicas de gerenciamento de sistemas Unix.

SUE faz poucas suposições sobre como e qual é o propósito de uma máquina utilizada no CERN (CERN / IT / PDP group, 2000). Estas máquinas são consideradas de uso geral e homogêneo, limitando às possibilidades de utilização e configurações específicas ou reservadas. Atualmente ele é limitado as seguintes arquiteturas: Digital UNIX; HP-UX; IBM AIX; GNU/Linux; e SUN Solaris.

A.3.6 NetView

NetView permite ao administrador da rede coletar informações estatísticas e gráficos, em tempo real, de muitos dispositivos. NetView é considerado o mais importante produto para a observação da disponibilidade dos dispositivos de uma rede. Ele estende o gerenciamento de rede tradicional para assegurar a disponibilidade de sistemas de negócio críticos e provê resolução rápida de problemas (HOCHSTETLER et al., 2000).

O NetView descobre a rede lógica. Endereços IP e informações da MIB de vários dispositivos são armazenados na sua base de dados. Utilitários organizam os elementos da rede em grupos comuns baseados em tipo de *hardware*, fabricante e outras características. Recursos do netmon (*polling* de *status* e configuração) pemirtem a identificação de falhas na rede. Além disso, é apresentado um relatório de eventos e possíveis causas de problemas na rede (HOCHSTETLER et al., 2000).

O NetView tem entre seus alvos o controle da complexidade de sistemas de comércio eletrônico de corporações (STAMME et al., 2000). Entre as características disponíveis para tanto estão (STAMME et al., 2000):

- a possibilidade de invocar qualquer comando TCP/IP, SNMP ou Unix através do NetView para sistemas OS/390. Isso pode ser feito por uma interface gráfica ou através de Clists que são escritas para estender as funcionalidades que o NetView disponibiliza;
- os serviços SNMP incluem um compilador, um carregador e um navegador de MIBs;
- interfaces Web de aplicação de terceiros podem ser lançadas pela ferramenta;
- instalação completa ou parcial de estações de trabalho (MCKECHNIE; NAKAJIMA; UELPENICH, 1995);
- monitoramento de eventos de *hardware*, emitindo alertas, eventos e dados estatísticos a fim de assistir na identificação de recursos problemáticos e na determinação de prováveis causas e ações recomendadas para problemas específicos a eventos de alerta (CASSAR et al., 1996);

NetView é uma ferramenta específica para o monitoramento e o gerenciamento de máquinas e sistemas da IBM. Possui uma boa gama de recursos e é útil para o gerenciamento de estações RISC de famílias como a RS6000 e gerenciamento de *software* do AIX.

O NetView é direcionado a produtos IBM gerenciáveis através de sub-sistemas de monitoramento e comandos SNMP. Não é um produto para uso genérico, flexível e extensível aos mais diversos tipos de sistemas e ambientes. Além disso, não é direcionado à geração e à manutenção de arquivos de configuração e atualização de serviços e sistemas de uma rede qualquer.

A.3.7 OpenView

OpenView é uma solução da HP para o gerenciamento de redes de computadores. Entre as aplicações da ferramenta podem ser citadas (Hewlett-Packard Development Company, L.P., 2004; Dell Inc., 1999):

- mapeamento e monitoramento de estados de ambientes de rede;
- administração de dispositivos de rede que tenham suporte ao protocolo SNMP;
- recursos para a navegação e acesso aos objetos de MIBs;
- consulta e atribuição de valores a objetos de MIBs;
- identificação de falhas e problemas de desempenho da rede, buscando atingir um diagnóstico;

HP OpenView é uma ferramenta comercial e tem entre seus princípios simplicidade, padronização e modularidade (Hewlett-Packard Development Company, L.P., 2004). É destinada a ambientes corporativos, buscando incrementar o desempenho da infraestrutura, antecipar e corrigir problemas e automatizar e gerenciar mudanças em tempo real (Hewlett-Packard Development Company, L.P., 2004).

O OpenView não é uma solução geral de gerenciamento para os mais diversos tipos de domínios administrativos. Além disso, carece de flexibilidade, adaptabilidade, independência de plataforma e gerenciamento integrado (instalação, configuração, manutenção, atualização) de serviços e sistemas de um modo mais abrangente.

A.3.8 FreeNMS

O FreeNMS (Grupo de Gerência de Redes - Laboratório Metropoa / PUCRS, 2004), é um sistema para o gerenciamento de redes. Entre suas funcionalidades podem ser citadas: monitoramento de redes, detecção de problemas e ativação de alarmes.

Este sistema realiza coletas periódicas de informações acerca dos equipamentos gerenciados. Os dados são armazenados em uma base de dados centralizada. Após a coleta e análise das informações armazenadas são gerados relatórios e alarmes.

O forte desta ferramenta é o monitoramento e a geração de alertas. Em sua atual versão não há suporte a uma base de dados centralizada com informações detalhadas sobre computadores (relacionáveis, hierarquizáveis e interconectáveis) e serviços de rede em geral. Também não existem recursos para a geração e posterior atualização de arquivos de configuração de máquinas e serviços. A auto-manutenção, dinâmica e automática, e a dinamicidade de gerenciamento de componentes e sub-sistemas são outros pontos fracos da ferramenta.

A.4 Ferramentas de Gerenciamento Local

A.4.1 Webmin

Webmin (CAMERON, 2004; WARRENE, 2004; SCHMIDT, 1999) funciona como uma interface abrangente e intuitiva para gerenciamento de aplicações em servidores. A configuração de aplicações e serviços como *ftp*, *ssh*, *mail*, *apache* e bancos de dados é uma das funcionalidades desta ferramenta.

Webmin executa em seu próprio mini-servidor Web e poderá estar disponível mesmo se o servidor HTTP padrão não estiver executando. É uma boa ferramenta para o gerenciamento remoto de alguns poucos servidores GNU/Linux. No entanto, gerenciar um grande número de servidores seria uma tarefa bastante custosa. Além disso, esta ferramenta não provê mecanismos de automatização de configurações.

Ela não é uma ferramenta de propósito geral e/ou que proporcione um ambiente de integração e relacionamento entre os parâmetros e dependências de configuração dos diferentes serviços.

A.4.2 SMIT

SMIT (SEGURA, 2000) do AIX (versão Unix da IBM) é uma alternativa a métodos típicos de utilização de comandos com sintaxes complexas, comuns do *shell* para o gerenciamento e manutenção das configurações do sistema operacional.

Esta ferramenta apresenta um menu (modo textual e/ou gráfico) interativo para o gerenciamento de sub-sistemas, dispositivos e processos do AIX. Com ela é possível realizar tarefas administrativas com maior rapidez e precisão.

O SMIT é uma ferramenta basicamente manual e destinada ao gerenciamento de uma única máquina. Além disso, não fornece uma base de informações integrada e dinâmica para o gerenciamento de serviços e sistemas.

A.4.3 LinuxConf

LinuxConf (GÉLINA, 2003) é uma ferramenta similar ao SMIT (seção A.4.2). Ela é basicamente manual e tem como objetivo facilitar o gerenciamento de um sistema GNU/Linux.

O LinuxConf é acompanhado de uma plataforma para a escrita de módulos administrativos, um grande conjunto de módulos e múltiplas interfaces (Web, textual e gráfica) de gerenciamento. Devido a isso, ela é uma ferramenta relativamente abrangente, para o gerenciamento de uma única máquina GNU/Linux.

O LinuxConf não provê base de informação centralizada, gerenciamento integrado de serviços e dispositivos de redes distribuídos pelo domínio administrativo, é pouco automatizado e pratica para o gerenciamento simultâneo de vários computadores.

A.4.4 CLIs

Outra maneira de configurar dispositivos de rede é através de interfaces de linha de comando (*CLIs - Command Line Interfaces*). Este é o meio mais comum para a manutenção de dispositivos e sistemas (LEE; CHOI, 2002). Isso se deve a vários fatores. Entre eles podem ser citados a codificação ASCII, a similaridade com *shells* Unix e orientação à seção, que normalmente está presente em dispositivos e é controlada por comandos (LEE; CHOI, 2002).

Estas interfaces de linha de comando não são adequadas para gerenciar uma rede com um grande número de equipamentos. Apesar da possibilidade de criação de *shell scripts* para a automação e agilização do processo de configuração e manutenção de dispositivos, é difícil pensar em administração de uma rede com dezenas — ou centenas — de computadores e outros periféricos através de interfaces de linha de comando (LEE; CHOI, 2002).

A.5 Outras Ferramentas

A.5.1 Sun Management Center

O centro de gerenciamento Sun (Sun Management Center) (Sun Microsystems, 2004) é um elemento para o monitoramento e gerenciamento de ambientes Sun. Este elemento possui recursos para unificar o gerenciamento de infraestrutura, provendo recursos como monitoramento de *hardware* e de aplicações do sistema.

Essa ferramenta é aberta, extensível no âmbito de monitoramento de sistemas e uma solução de gerenciamento desenvolvida em Java, que utiliza o protocolo SNMP para possibilitar o gerenciamento de dispositivos de rede que tenham suporte a este protocolo.

Alguns recursos disponíveis:

- monitoramento avançado do Solaris e de *hardware* da SUN;
- gerenciamento de *patch*, monitoramento de arquivos, carregador de *scripts* e analisador de falhas do sistema operacional Solaris;
- gerenciador de relatórios de desempenho de produtos SUN gerenciáveis;
- gerenciador de atualizações de sistemas, permitindo instalações e atualizações com o sistema em funcionamento;
- gerenciador de disponibilidade de serviços, monitoramento e confirmação da disponibilidade de serviços, como provedores de serviços Web, diretórios FTP, e-mail e o calendário de serviços do Solaris.

O centro de gerenciamento Sun é destinado a ambientes Sun, e não compõe uma solução abrangente para o gerenciamento de redes de computadores. É restrito a operações de protocolos como o SNMP, falhando em escalabilidade, flexibilidade, segurança e adaptabilidade.

A.5.2 Ferramentas Diversas

Existem várias outras ferramentas disponíveis e em uso para o monitoramento de redes de computadores. Entre elas podem ser citadas: SNMPMonitor (SCHAUMANN, 2001), PSWeM (AZAMBUJA, 2001), GFI Network Server Monitor (GFI Software Ltd., 2004), IPCheck Server Monitor (Paessler GmbH, 1998), NetCrunch (AdRem Software, Inc., 2004), WhatsUp (Ipswitch, Inc., 2004), Alchemy Eye (Alchemy Lab, 2004), NetCentral (John Maler Thomson, 2003) e Big Brother (Quest Software, 2002). Muitos destes aplicativos são de uso exclusivo para o monitoramento de computadores e dispositivos de rede em geral. Outros possuem alguma habilidade gerencial sobre o protocolo SNMP. Neste último grupo entram ferramentas como o PSWeM e NetCrunch. Estas duas têm uma aplicação gerencial semelhante a FreeNMS (seção A.3.8), ou seja, estão restritas às operações gerenciais do protocolo SNMP.

A.5.3 DCAST

DCAST (Doshisha Cluster Auto Setup Tool) (HIROYASU et al., 2003) é um sistema que facilita a construção de aglomerados de computadores de larga escala. Esta ferramenta funciona em conjunto com a distribuição GNU/Linux, do Debian. Seu principal objetivo é facilitar a vida de administradores inexperientes na construção e atualização de aglomerados de computadores.

O funcionamento básico deste sistema reside na abstração de operações e conhecimentos necessários para instalar um sistema operacional, manter consistência de *software* entre nós individuais, *software* que precisa de atualização e aspectos de segurança (HIROYASU et al., 2003).

Em suma, esta ferramenta tem uma aplicação bastante específica e restrita. O uso de seus recursos em uma rede de computadores heterogêneos seria impraticável. Além de não prover mecanismos mais sofisticados e flexíveis para o gerenciamento de aplicações e computadores.

A.5.4 FAI

FAI (Fully Automatic Installation) (LANGE, 2004) é uma ferramenta para a instalação automática de uma máquina ou um conjunto inteiro de computadores. Este sistema é baseado na distribuição GNU/Linux do Debian.

O público-alvo do FAI são administradores de sistemas que precisem instalar o Debian em um ou até centenas de computadores (LANGE, 2004). No entanto, esta ferramenta é exclusivamente destinada a instalação de sistemas Debian. Contudo, a manutenção e refinamento de configurações de serviços e computadores devem ser feitos utilizando outros recursos.

A.5.5 Update-Cluster

Update-Cluster (UEKAWA, 2003) é uma ferramenta que prevê a unificação dos arquivos de configuração de um sistema. Através desta ferramenta é possível unificar o controle de informações relativas a uma máquina, que são freqüentemente utilizados por aplicativos relacionados a aglomerados de computadores. O objetivo é poder facilmente duplicar a informação em vários sistemas ou aplicações através de uma solução baseada em XML.

O ambiente de manutenção de configurações de aplicações disponibilizado pelo Update-Cluster é baseado em um modelo centralizado de armazenamento e manipulação de diferentes arquivos de configuração. Estes arquivos são mantidos em uma máquina-mestre e a partir desta distribuídos às máquinas clientes.

As informações de configuração são mantidas em um arquivo XML e a partir deste são gerados os arquivos de configuração para diferentes aplicações. Este modelo de sistema é bastante útil para a manutenção e a cópia de arquivos de configuração gerais. Porém, um recurso desse gênero não fornece flexibilidade, liberdade e escalabilidade para domínios com ambientes diversificados ou domínios com um número grande de computadores a serem gerenciados. Não obstante, ele não foi projetado para ser uma ferramenta que permite gerenciamento integrado (com dependências de dados, atualização automática de configurações de aplicações e distribui-

ção de dados e carga de processamento) de sistemas.

A.5.6 OSCAR

OSCAR (Open Source Cluster Application Resources) (LIGNERIS et al., 2003) é um conjunto de ferramentas que permitem uma fácil e rápida instalação e manutenção de aglomerados de computadores. O principal objetivo é integrar em um único pacote aplicativos e sistemas necessários para instalar e gerenciar conjuntos de máquinas destinadas ao processamento de alto desempenho.

OSCAR não é um sistema de administração de máquinas para diferentes domínios administrativos. Ele é uma solução para usuários que pretendem criar aglomerados de computadores, principalmente para a execução de aplicações científicas, baseados nas distribuições GNU/Linux RedHat e Mandrake. Uma das principais intenções é simplificar e facilitar a vida de usuários inexperientes ou que não necessitem ambientes específicos, enxutos e otimizados para uma determinada arquitetura de máquinas.

A.5.7 NPACI Rocks

NPACI Rocks (PAPADOPOULOS; KATZ; BRUNO, 2003) é um sistema semelhante ao OSCAR (seção A.5.6). Rocks é formado por uma distribuição GNU/Linux e conjuntos de ferramentas que agilizam e facilitam a instalação e manutenção de aglomerados de computadores.

Rocks, ao contrário de ferramentas como o Cfengine, realiza re-instalações completas do sistema operacional em elementos que precisam ser atualizados no aglomerado. A idéia por trás deste processo é que pode ser mais rápido e eficiente uma instalação completa do que a análise e atualização de aplicativos individualmente (PAPADOPOULOS; KATZ; BRUNO, 2003).

Devido a filosofia do sistema Rocks, além de seu domínio específico de aplicação, ele não é aplicável a redes de computadores de um modo geral. Ambientes heterogêneos e servidores de rede que não podem parar são alguns dos componentes de rede que não poderiam ser mantidos e gerenciados por um sistema integrado (um verdadeiro pacote fechado, que possui uma receita de aplicação e funcionamento pré-definida) como o Rocks.

A.5.8 Ka Clustering Tools

Ka (AUGERAT, 2003) é um conjunto de ferramentas de código aberto desenvolvidas para auxiliar na instalação e utilização de aglomerados de computadores. Grosseiramente, é um sistema semelhante ao Rocks (seção A.5.7) e ao OSCAR (seção A.5.6).

Os ambientes de aplicação do Ka são máquinas com GNU/Linux ou Windows. Um dos pontos fortes do sistema é a escalabilidade (AUGERAT et al., 2002; AUGERAT, 2003). O Ka foi desenvolvido para aglomerados de computadores de larga escala, com dezenas, centenas ou até milhares de computadores. O funcionamento básico reside na clonagem de sistemas referência. Além disso, o Ka também prove alguns recursos para o gerenciamento de processos, ativação e manutenção de um aglomerado de computadores.

Contudo, o Ka não é um ambiente adequado e propício ao gerenciamento de redes de computadores. Ele é um sistema para domínios específicos, com aplicações e requisitos delimitados e reduzidos.

APÊNDICE B BASE DE DADOS

Este apêndice apresenta um exemplo de uma relação simples entre dados da base de dados e alguns exemplos de entradas cadastradas no sistema.

B.1 Exemplo de uma Relação Simples na Base de Dados

Abaixo segue um exemplo de configuração de cinco atributos do serviço de DNS. O exemplo apresenta 3 atributos com relações de dependência e dois sem.

```
1 dnsAdminMailbox: cn=HostMaster,ou=Geral,dc=domínio,dc=br
2 dnsZoneMaster: cn=máquina01,ou=Máquinas,dc=domínio,dc=br
3 sOARecord: dns.domínio.br
4 mXRecord: cn=SMTPprimário,ou=Mapeamentos,dc=domínio,dc=br
5 nSRecord: 192.168.0.1
```

Pode-se perceber que são referenciadas informações de três diferentes sub-árvores da base de dados. Na linha 1 é feita uma referência a uma entrada geral que define qual é o endereço do *hostmaster* do domínio. Em uma organização consistente do sistema, essa entrada deveria ser referenciado por todos os agentes e módulos que gerarão alguma informação ou arquivo de configuração que use o endereço do *hostmaster*. Pois, caso o *hostmaster* mude, todos os sistemas ou serviços serão atualizados automaticamente com o novo endereço. Isso é válido também para as demais referências dos atributos seguintes.

No exemplo da linha 2 é feita uma referência a máquina01 da base de dados. No entanto, a definição do servidor primário de zona do DNS poderia ser feita em um mapeamento. Neste caso, todas as sub-configurações de serviços ou sistemas apontariam para o mapeamento. Quando o servidor primário de zona mudar, basta acertar a entrada do mapeamento que todos os serviços e sub-sistemas seriam adequadamente atualizados. Este é o caso da referência na linha 4.

Nos casos das linhas 3 e 5 são utilizados o nome completo da máquina e o endereço IP, respectivamente. Essa é uma opção disponível ao administrador do sistema. Contudo, este deve estar ciente dos possíveis futuros riscos de configurações errôneas de serviços ou sistemas devido a uma possível mudança de máquina responsável pelo serviço de DNS.

Todas as opções são possíveis. Cabe ao administrador da rede decidir como organizar e utilizar a informação disponível.

O `ida`, em sua versão atual, usa a representação própria de entradas LDAP para estabelecer relacionamentos. Para usuários leigos, não inteirados com o LDAP, pode parecer algo complicado e difícil. Por outro lado, para usuários que, por exemplo, lidam (ou manipularam) com diretório LDAP para a autenticação de usuários (o que esta se tornando cada vez mais comum em domínios administrativos tanto corporativos quanto acadêmicos) isso é algo normal, intuitivo e lógico. A idéia foi não criar uma nova linguagem ou conjuntos de expressões para a representação de relacionamento, o que não seria uma tarefa difícil. Se um novo conjunto de padrões de representação fosse criado os usuários seriam obrigados a estudar e compreender uma forma de referência e relacionamento de dados. Isso não seria uma política interessante, principalmente em administração de redes, onde já existem muitos sistemas, ferramentas e recursos que precisam ser compreendidos, configurados e mantidos e, por isso, acabam tomando muito tempo dos gerentes da rede.

No caso do `ida`, o administrador tem disponível um pequeno pacote de *software* que lê, interpreta e resolve as relações e dependências de uma entrada da base de dados. Isso facilita o trabalho de desenvolvimento de agentes e módulos que fazem uso das informações contidas na base de dados.

B.2 Exemplos de Entradas

Esta seção apresenta exemplos de entradas cadastradas na base de dados do `ida`. Os exemplos fornecem uma melhor noção da representação interna (da base de dados) dos dados de máquinas, serviços e outras características e recursos do sistema. Os atributos iniciados por `inms` são do `ida`, enquanto que os demais são de esquemas definidos e mais tempo e disponíveis para o uso. Logo, uma das intenções do sistema foi manter compatibilidade com sistemas existentes.

B.2.1 Registro de uma Máquina

No caso da figura B.1, o atributo `inmsHostNetworkConfig` especifica que existe uma herança de configurações. Estas configurações gerais de rede incluem dados como o endereço do servidor DNS, endereço do *gateway* da rede, nome do domínio, máscara da rede e endereço da rede. Caso algum destes dados não esteja presente na entrada da máquina, será acrescentado no momento em que a entrada do computador for processada.

Outros atributos específicos, como `inmsMachineCPU`, `inmsMachineMemory`, `inmsMachineHD`, `inmsMachineNetworkCard` e `inmsMachineMotherboard` podem ser utilizados. Esse seria o caso de um domínio administrativo que deseje ter um relatório completo e detalhado de todos os equipamentos da rede, por máquinas completas e por componentes internos.

```
dn: cn=computador01.domínio.br,ou=Máquinas,dc=domínio,dc=br
objectClass: inmsNetworkHost
objectClass: inmsGeneral
cn: computador01.domínio.br
inmsIpHostName: computador01
inmsIpDomainName: domínio.br
ipHostNumber: 10.129.179.8
ipHostNumber: 10.129.179.15
macAddress: 00:DD:CC:B1:A0:FD
inmsHostAlias: servidor1
inmsHostAlias: webmail
inmsService: ftp
inmsService: ssh
inmsOperatingSystems: Gentoo GNU/Linux
inmsMachineDescription: Dual-Pentium-III, 1GB de RAM, 200GB de Disco
inmsMachineLocation: Sala-777
inmsGeneralInformation: Máquina velha.
inmsMachinesGroup: GrupoDNS1
inmsHostNetworkConfig: cn=ConfiguraçãoDaRede,ou=Geral,dc=domínio,dc=br
inmsLastEntryUpdate: 9
```

Figura B.1: Exemplo de registro de uma máquina

B.2.2 Registro de um Serviço

Os dados apresentados na figura B.2 representam parte da configuração de um serviço de DNS padrão de uma rede local.

```
dn: cn=DNSprimário,ou=Serviços,dc=domínio,dc=br
cn: DNSprimário
objectClass: top
objectClass: dnszone
objectClass: inmsDomain
objectClass: inmsGeneral
objectClass: inmsService
ipServiceProtocol: tcp
ipServiceProtocol: udp
ipServicePort: 53
dnszonename:.domínio.br
dnsserial: 1095255343
dnsrefresh: 3600
dnsretry: 900
dnsexpire: 1209600
dnsttl: 43200
dnsminimum: 86400
dnsadminmailbox: hostmaster.domínio.br
dnszonemaster: dns.domínio.br
dnstimestamp: 1095255343
sOARecord: dns.domínio.br
mXRecord: smtp.domínio.br
nSRecord: dns.domínio.br
aRecord: loopback IN A 127.0.0.1
aRecord: localhost IN A 127.0.0.1
aRecord: fwinf IN A 200.132.35.10
inmsDNSrevzPTR: 2 IN PTR mss001.domínio.br
inmsDNSrevzPTR: 3 IN PTR mss002.domínio.br
inmsMyMachines: GrupoDNS1
inmsEntryDependency: cn=ConfiguraçãoDaRede,ou=General,dc=domínio,dc=br
inmsBranchDependency: ou=Máquinas,dc=domínio,dc=br
inmsServiceStoreEntry: cn=DNSprimário,ou=ConfigFiles,dc=domínio,dc=br
inmsConfigurationModuleEntry: cn=DNSprimário,ou=MódulosDeGeração,dc=domínio,dc=br
inmsUpdatingModuleEntry: cn=DNSprimário,ou=MódulosDeAtualização,dc=domínio,dc=br
inmsServiceDirectory: /etc/bind
inmsLastEntryUpdate: 2
```

Figura B.2: Exemplo de uma entrada para o serviço de DNS

Cada atributo de sua entrada faz parte da configuração final do serviço de DNS.Os atributos

apresentados na figura B.3 representam relações de dependência de dados (quando algum destes dados for atualizado o módulo de geração do DNS primário será automaticamente ativado), localização dos arquivos de configuração finais e dos módulos de configuração e atualização do serviço e o diretório padrão para a atualização dos arquivos de configuração do serviço.

- inmsEntryDependency
- inmsBranchDependency
- inmsServiceStoreEntry
- inmsConfigurationModuleEntry
- inmsUpdatingModuleEntry
- inmsServiceDirectory

Figura B.3: Atributos de relação de dependência de dados

O atributo `inmsMyMachines` define qual é o grupo de máquinas que fará parte da configuração do DNS. Neste caso, será o grupo `GrupoDNS1`, ou seja, todas as máquinas pertencentes a este grupo receberão uma entrada na tabela de resolução de nomes do DNS. A máquina apresentada na configuração da seção B.2.1 é um dos computadores que fazem parte deste grupo.

B.2.3 Registro de um Mapeamento

O mapeamento apresentado na figura B.4 corresponde a identificação da máquina que é o servidor primário do DNS. A entrada de mapeamento identifica a referência aos servidores primário e secundário na árvore de máquinas. Poderia ainda constar uma referência a um terceiro servidor. O servidor válido sempre é o primário. Os demais servem para o caso de migrações ou tentativas de recuperação de serviço.

```
dn: cn=DNSprimário,ou=Mapeamento,dc=domínio,dc=br
objectClass: top
objectClass: inmsMapping
cn: DefaultNetDNS
inmsPrimaryServer: cn=servidor01.domínio.br,ou=Máquinas,dc=domínio,dc=br
inmsSecundaryServer: cn=servidor02.domínio.br,ou=Máquinas,dc=domínio,dc=br
inmsServiceEntry: cn=DNSprimário,ou=Serviços,dc=domínio,dc=br
inmsFaultPolicy: tryUpdateService1 and alertManager
inmsFaultPolicy: tryUpdateService2
inmsFaultPolicy: tryMigrateService
inmsFaultPolicy: tryUpdateService3
inmsLastEntryUpdate: 1
```

Figura B.4: Exemplo de mapeamento

Os quatro valores do atributo `inmsFaultPolicy`, que define políticas a serem tomadas no caso de o serviço não mais estar respondendo, são regras a serem seguidas pelo agente responsável pela análise dos resultados de monitoramento de serviços. No caso da configuração apresentada, o agente tentará, em uma primeira instância, acionar o módulo de geração para provocar uma atualização do serviço de DNS e alertar o administrador do sistema. Caso o serviço continue apresentando falhas de funcionamento, uma segunda tentativa de atualização

será forçada a acontecer. A terceira opção é migrar o serviço. Neste caso, o servidor secundário passará a ser o primário e este passará a ser o secundário, ou seja, haverá uma inversão de papeis entre os servidores. Caso continue não funcional, uma nova tentativa de atualização, no novo servidor primário, será executada. Caso nenhuma das tentativas tenha sucesso, cabe ao administrador do sistema averiguar, encontrar e resolver o suposto problema.

B.2.4 Registro de *Log*

```
dn: cn=20041029145048.512116.7370.18883-20041029145110.740639.19404.18883,cn=
 relho#192.168.179.79,cn=DNSmonitor.MonitoringAgents.domínio.br,ou=Relatórios
 DeMonitoramento,dc=domínio,dc=br
cn: 20041029145048.512116.7370.18883-20041029145110.740639.19404.18883
objectClass: top
objectClass: inmsGlobalLog
inmsLog:: MjAwNDEwMjkxNDUwNTguNzIwNDY2LjcwOTUxLjA3NDIyNTkzNTEuMTg4ODMjMSMjIyNb
 Li9ETlNtb25pdG9yLk1vbml0b3JpbmdBZ2VudHMucGxhbmV0LmVhcnRoLnBlLnBsOjc5XSA9PiBXQ
 VJJTklORzogU0VSVklDRSBOT1QgUkVQT05ESU5HIQo=
inmsLog:: MjAwNDEwMjkxNDUxMTAuNzQwMjExLjE0ODg4LjAyODEzNDI3NzcuMTg4ODMjMiMjIyNb
 Li9ETlNtb25pdG9yLk1vbml0b3JpbmdBZ2VudHMucGxhbmV0LmVhcnRoLnBlLnBsOjg5XSA9PiBXQ
 VJJTklORzogSE9TVCBOT1QgUkVTUE9OREl0RyEK
```

Figura B.5: Exemplo de um registro de *log*

Os dados da figura B.5 representam uma entrada de *log* da base de dados do `ida`. A entrada (`dn`) é identificada pelo hora (no formato: ano, mês, dia, horas, minutos e segundo - `20041029145048`), por micro-segundos do tempo atual (`512116`), por um número aleatório (de zero a 100000 - `7370`) e por um identificador de processo (`18883`). O identificador final da entrada de *log* é composto por duas cadeias de caracteres, contendo os dados descritos anteriormente, separadas pelo marcador '**-**'. O nome da máquina, o endereço IP e o nome do módulo que está registrando os dados de *log* também fazem parte da identificação da entrada na base de dados. O objetivo dessa extensa nomenclatura (da entrada - `dn`) é evitar entradas com a mesma identificação e facilitar a consulta por informações específicas de atividade dos componentes do sistema. A seção D.5, do apêndice D, apresenta uma descrição da ferramenta de análise e uma visualização de informações disponíveis nos *logs* do sistema.

Os registros de *log* são armazenados nos atributos `inmsLog`. Eles são codificados em base 64. Isso por que contêm tipos e definições variadas de dados. Os *logs* podem conter cadeias de caracteres, números inteiros, números reais e qualquer forma de dados expressáveis pelas funções padrão de impressão, disponíveis nas linguagens de programação mais comuns em sistemas Unix - como C, Perl, *Shell Scripts* e Python.

B.2.5 Registro de *Timing*

Os registros de temporização (exemplo na figura B.6) têm uma composição de identificação semelhante ao registros de *log* (ver seção B.2.4). A excessão é o atributo que armazena os dados. Este contém os registros de temporização em formato de texto, como pode ser visto no exemplo apresentado nesta seção.

Cada atributo de temporização tem basicamente duas coisas: uma identificação e um in-

```
dn: cn=20050006100829.947493.21803.29375-20050006100833.8330.25850.29375,cn=co
 mpig.domínio.br#192.168.179.88,cn=Server1.ServicesAgents.domínio.br,ou=2maqui
 nas6agentes,dc=domínio,dc=br
cn: 20050006100829.947493.21803.29375-20050006100833.8330.25850.29375
objectClass: top
objectClass: inmsTiming
inmsLog: 1 #### [TAG:] MAINPOLLINGTIME 1024 [VALUE:] 1105013309.995628-1105013
 309.995710
inmsLog: 2 #### [TAG:] search 1 [VALUE:] 1105013309.995823-1105013310.17113
inmsLog: 3 #### [TAG:] search 2 [VALUE:] 1105013310.18033-1105013310.42352
inmsLog: 4 #### [TAG:] system exec ./inmsModules.Server1/NetworkDNS.Configurat
 ionCode.inf.ufsm.br.pl [VALUE:] 1105013310.43340-1105013312.941790
inmsLog: 5 #### [TAG:] update 2 [VALUE:] 1105013312.942165-1105013312.942381
inmsLog: 6 #### [TAG:] search 4 [VALUE:] 1105013312.950176-1105013312.969599
inmsLog: 7 #### [TAG:] search 5 [VALUE:] 1105013312.970615-1105013312.981782
inmsLogAlreadyProcessed: 1
```

Figura B.6: Exemplo de um registro de temporização

tervalo de tempo. A identificação determina o que foi temporizado (no exemplo apresentado, indicado pela `[TAG:]`, exemplo: `search 1`) e o tempo de duração (tempo inicial-final em segundos, exemplo: `1105013309.995823-1105013310.17113`).

A seção D.4 do apêndice D apresenta um exemplo simples de uma visualização dos dados de temporização armazenados na base de dados.

APÊNDICE C FERRAMENTAS

C.1 Linguagens Utilizadas

C.1.1 Perl

Perl (Practical Extraction and Reporting Language) (The Perl Foundation, 2002) é uma linguagem de programação que tem por objetivo transformar trabalhos fáceis em fáceis, sem fazer com que os trabalhos difíceis sejam impossíveis. Esta linguagem facilita a manipulação de números, textos, arquivos, diretórios, computadores, redes e programas. Ela também facilita o desenvolvimento, modificação e depuração de seus próprios e portáveis programas, em qualquer sistema operacional moderno (SPAINHOUR; SIEVER; PATWARDHAN, 2002; CHRISTIANSEN; TORKINGTON, 1998). Além disso, Perl possui grande parte dos recursos e ferramentas que outras linguagens modernas possuem, como referências, orientação a objetos, compiladores, interpretadores, depuradores (FOLEY, 2003), rastreadores, referências cruzadas, editor de sintaxe, bibliotecas, recursos de programação para Web (BURKE, 2002), redes (STEIN, 2000) e BioInformática (TISDALL, 2003).

Perl também é uma linguagem simples e bastante rica. É uma linguagem simples de modo que as estruturas de dados são simples de serem utilizadas e compreendidas. Ela é rica por que fornece liberdade, flexibilidade e recursos que permitem aos usuários resolverem problemas difíceis (SPAINHOUR; SIEVER; PATWARDHAN, 2002; SCHWARTZ; PHOENIX, 2003).

C.1.2 Java

Java é uma linguagem moderna que herda aspectos de sintaxe da linguagem C e de orientação a objetos da linguagem C++ (NAUGHTON; SCHILDT, 1999). É uma linguagem livre de plataforma e apropriada para o desenvolvimento de uma variada gama de soluções tecnológicas (KNOERNSCHILD, 2001; DARWIN, 2001; DEITEL; DEITEL; SANTRY, 2001).

O `ida` suporta a linguagem Java através de ambientes JNDI como o conjunto de ferramentas de desenvolvimento da Novell (Novell, Inc., 1998a), o ambiente de desenvolvimento de programas da Mozilla (TSAI, 2004) e o driver para JDBC da OctetString (OctetString, Inc., 2004). Atualmente o `ida` conta com alguns recursos desenvolvidos em Java.

C.1.3 C

A linguagem C (KERNIGHAN; RITCHIE, 1988; LINDEN, 1994) é conhecida por sua vasta aplicação em desenvolvimento de sistema operacionais e aplicações de baixo nível [1]. Esta é uma linguagem eficiente em termos de desempenho se comparada a linguagens como Java e Perl. No entanto, possui um ciclo de desenvolvimento maior e, normalmente, um menor reaproveitamento de código. Além de ser compilada, ou seja, sua portabilidade é bastante prejudicada em relação a linguagens interpretadas.

No caso do `ida`, existem basicamente três ambientes de desenvolvimento para a linguagem C. Na verdade, são bibliotecas e aplicativos fornecidos pela Novell (Novell, Inc., 1998b), Mozilla (SMITH, 1998; SMITH et al., 2000) e OpenLDAP (OpenLDAP Foundation, 2004a). Logo, o `ida` tem suporte, além de ferramentas implementadas, a linguagem C.

C.1.4 *Shell Scripts*

Shell scripts são recursos bastante conhecidos e difundidos entre administradores de sistemas. Os motivos vão desde o uso para resolver problemas simples e momentâneos até o desenvolvimento de ferramentas básicas e avançadas para o gerenciamento de sistemas. A portabilidade, praticidade e desenvolvimento rápido estão entre alguns dos fatores que também levam gerentes de sistemas ao uso de *shell scripts* (COOPER, 2003; GITE, 2002; MARTIN, 2001; ALBERT, 1999; OUSTERHOUT, 1998).

O `ida` possui ferramentas e módulos desenvolvidos em linguagem *shell scripts*. O desenvolvimento destes componentes foi baseado no Bash (COOPER, 2003). Estes *shell scripts* utilizam recursos de conexão a base de dados de conjuntos de ferramentas como as providas pelo OpenLDAP (OpenLDAP Foundation, 2004a) e Ldaptor (VIRTANEN, 2003a). Nas atuais implementações do `ida` são utilizadas as ferramentas providas pelo OpenLDAP.

C.1.5 Python

Python (CHUN, 2000; LEMBURG, 2004) é uma linguagem interpretada semelhante ao Perl. Ela é uma linguagem para a programação Internet e de sistemas. Uma de suas grandes características é ser um ambiente para o rápido desenvolvimento de aplicações, ser simples, orienta da objetos, extensível, escalável e com uma sintaxe concisa e simples de entender (CHUN, 2000; BEAZLEY, 2000; O'Reilly & Associates, 2003).

O `ida` tem suporte a Python. Além disso, contém recursos implementados nessa linguagem.

C.1.6 PHP

PHP (ATKINSON, 2003) é uma linguagem de programação para a Internet. Ela é simples (LERDORF; TATROE, 2002; ZANDSTRA, 2003) e tem, ao mesmo tempo, o propósito básico

[1] Baixo nível, neste texto, significam aplicações que possivelmente seriam desenvolvidas em Assembly. Programas neste nível incluem aplicativos e interfaces para o controle de componentes eletrônicos de microcomputadores e equipamentos tecnológicos em geral, como impressoras fiscais, PDAs, tocadores MP3

de proporcionar um ambiente poderoso de desenvolvimento para a Web (LERDORF; TATROE, 2002; ATKINSON, 2003). Esta linguagem proporciona a exibição de conteúdos HTML através de recursos de programação de linguagem sofisticadas como C e *scripts* CGI (Common Gateway Interface) (ATKINSON, 2003).

PHP também é uma das melhores alternativas para o desenvolvimento Web em geral (ATKINSON, 2003; LERDORF; TATROE, 2002). Esta linguagem é melhor que as demais, mais rápida para codificar e executar, o mesmo código (sem nenhuma alteração) roda em diferentes servidores Web e diferentes sistemas operacionais, é livre, é modificável, entre outras características e recursos (ATKINSON, 2003; LERDORF; TATROE, 2002).

Esta linguagem tem suporte a LDAP (The PHP Group, 2004), logo, pode ser utilizada para gerenciar, melhorar ou incrementar o `ida`. Além disso, existem interfaces de navegação em diretórios LDAP escritas em PHP que podem ser utilizadas para visualizar e gerenciar a base de dados do `ida` (ver seção C.3.3).

C.2 Linguagens Suportadas

Além das tecnologias utilizadas (seção 4.2) nesta seção são apresentadas algumas tecnologias, mais especificamente linguagens de programação, atualmente suportadas no sistema.

C.2.1 Objective-C

Objective-C é uma linguagem de programação orientada a objetos baseada na linguagem C. Esta linguagem introduz um dos mais extensivos ambientes de desenvolvimento orientado a objetos atualmente disponíveis (HILLEGASS, 2002; ANGUISH; BUCK; YACKTMAN, 2003).

O SOPE (SKYRIX Software AG, 2004) é um extensivo conjunto de ferramentas que formam um ambiente completo para aplicações Web(SKYRIX Software AG, 2004). Neste ambiente esta incluso o SOPE-LDAP, que é um conjunto de bibliotecas para o gerenciamento de diretórios LDAP em Objective-C (SKYRIX Software AG, 2004).

C.2.2 Ch LDAP

Ch LDAP (SoftIntegration, Inc., 2004a) é um pacote, para o ambiente de desenvolvimento Ch (SoftIntegration, Inc., 2004b), que permite a implementação de sistemas que façam uso do LDAP. O Ch é um interpretador C/C++ para programação estilo *scripting*, programação *shell*, computação numérica (SoftIntegration, Inc., 2004b).

C.2.3 Delphi

A linguagem Delphi, da Borland, é conhecida como um bom ambiente para o desenvolvimento de aplicações cliente-servidor e aplicativos em geral para plataformas Microsoft Windows (CANTù, 2001). As virtudes do Delphi vão de completo suporte a programação orientada a objetos ao suporte de desenvolvimento visual através da combinação de sofisticação e facili-

dade (CANTù, 2001; Borland, 2002).

Delphi também tem suporte a LDAP através de componentes de desenvolvimento (BURKE, 2001; n software inc., 2004; SCHREIBER, 2002). Devido a isso, essa linguagem também figura entre as linguagens que podem ser utilizadas para o desenvolvimento, melhoramento, extensão ou adaptação do `ida`.

C.3 Tecnologias Utilizadas

C.3.1 LDAP

O LDAP (Lightweight Directory Access Protocol) (YEONG; HOWES; KILLE, 1995; WAHL; HOWES; KILLE, 1997; HODGES; MORGAN, 2002; KOUTSONIKOLA; VAKALI, 2004) é um padrão da indústria que vem evoluindo para suprir necessidades do comércio eletrônico no que diz respeito a aplicações distribuídas, representação da informação na forma de diretórios *online*, diretórios que evitem a criação de ilhas de informação, entre outras coisas (JOHNER et al., 1999).

O alvo do LDAP é a informação que cabe a um diretório *online*, como nomes, endereços, senhas, autorizações de acesso, linguagem nativas, entre outras coisas (LAIRD; SORAIZ, 1999). Em suma, o LDAP é apenas um protocolo de rede. Este protocolo pode ser alimentado por diferentes bases de dados (ver exemplos na seção C.3.2).

Entre as características do protocolo LDAP podem ser citadas:

Segurança. Como LDAP está cada vez mais se tornando um padrão para a indústria (JOHNER et al., 1999; LAIRD; SORAIZ, 1999; SOLBRIG; CHUTE, 2004; CASTAN; BASTIEN, 2003), segurança é um dos pontos que precisam ser fortemente atacados. Esta tecnologia é atualmente utilizada para o armazenamento *online* de informações acerca de corporações e seus clientes, logo, precisa ser confiável e segura. Devido a isso, esta nova tecnologia, desde sua modelagem e concepção, aborda e implementa variados aspectos de segurança da informação (BAO, 2001; BIALASKI, 2000; WEBER, 2002; HODGES; MORGAN; WAHL, 2000; MYERS, 1997; STOKES et al., 2000; HOWES; SMITH, 1997; WAHL et al., 2000; ZEILENGA, 2001b; CASTAN; BASTIEN, 2003).

Replicação nativa. Esta característica é útil para aumentar a disponibilidade do diretório *online* e para realizar cópias de segurança da base de dados (JOHNER et al., 1999; University of Michigan, 1996; STOKES et al., 2002).

Escalabilidade/distribuição. No LDAP os dados podem facilmente ser distribuídos entre servidores de dados (University of Michigan, 1996; HOWARD, 1998; GULBRANDSEN; VIXIE; ESIBOV, 2000), além da replicação nativa. Os servidores LDAP irão procurar a informação, solicitada pelo usuário, na árvore de informação do diretório (DIT - Directory Information Tree) (JOHNER et al., 1999; GULBRANDSEN; VIXIE; ESIBOV, 2000). O

funcionamento é semelhante ao do DNS, utilizando recursos de hierarquia e roteamento (KILLE, 1995a, 1998, 1995b).

Dinamicidade. O LDAP suporta entradas de dados dinâmicos (YAACOVI; WAHL; GENOVESE, 1999). Isso significa que podem existir dados que estão momentaneamente na base de dados e podem desaparecer a qualquer hora. Esse poderia ser o caso de controle/relatório dinâmico de usuários *online* (com suas características, preferências e configurações) em um sistema Web. Enquanto o usuário está conectado e ativo sua entrada (dinâmica) de dados esta presente. No momento em que o usuário sai de contexto, a entrada some da base de dados. Essa ilustração demonstra que o LDAP é dinâmico, assim como a MIT (Management Information Tree) do CMIP (Management Information Protocol) da OSI (Open Systems Interconnection) (YEMINI, 1993), sendo ambos superiores e mais eficientes para manipulações, representação e armazenamento de dados, em relação as MIBs SNMP (YEMINI, 1993).

Hierarquia. Além da hierarquia de escalabilidade e distribuição da árvore de informação do diretório, existe também a hierarquia organizacional dos dados propriamente ditos (JOHNER et al., 1999). A base de dados LDAP pode ser organizada hierarquicamente, com diferentes níveis de acesso e relações de dependência. Essas características auxiliam na criação e manutenção de grandes e diversificadas quantidades de dados.

Descrição de informação semi-estruturada. Com LDAP fica a cargo do usuário a definição do esquema (WAHL, 1997; FLEMING; MCDONALD, 2004; WAHL, 2000; ZEILENGA, 2001c; GREENBLATT; RICHARD, 1999) e da forma como os dados serão organizados (JOHNER et al., 1999; HOWES, 1999; JOHNER et al., 1998; GOOD, 2000; KOUTSONIKOLA; VAKALI, 2004). Outra decisão a cargo do usuário é o tipo de dados que serão armazenados na base de dados. Cada atributo (WAHL et al., 1997; WAHL; HOWES; KILLE, 1997) do esquema LDAP pode suportar desde dados em texto puro até informação em formato binário.

Desempenho. Para sítios grandes, LDAP pode ser mais eficiente que arquivos puros (WILLIAMS, 2003; WANG et al., 2000). Em relação a protocolos como o NIS, o LDAP apresenta ganhos consideráveis de desempenho, com pacotes menores e tempos de resposta melhores (BUENO; CARISSIMI, 2004). Diversos aspectos e resultados de desempenho do LDAP, em ambientes e especificidades variados, foram reportados (WANG et al., 2000; HOWES; SMITH, 1995; THONTON; MUNDY; CHADWICK, 2003; DIXON et al., 2002; ISQUIERDO, 2001). Os resultados, de um modo geral, apontam o LDAP como uma boa solução para a manipulação e controle de diretórios *online*.

Granularidade administrativa. Outra característica do LDAP é a possibilidade de definir níveis administrativos. Um administrador, ou um determinado DN (Distinguished Name),

poderá ter acesso refinado, em nível de atributo, indo até o acesso global a árvore de informação do diretório (University of Michigan, 1996; JOHNER et al., 1999, 1998; HOWES, 1999; WILLIAMS, 2003). Com isso pode-se estabelecer vários níveis de controle, acesso e gerenciamento.

Proteção contra erros tipográficos e sintáticos. Esquemas e atributos bem definidos (WAHL; HOWES; KILLE, 1997; WAHL, 1997; WAHL et al., 1997) evitam que erros tipográficos e sintáticos sejam incluídos na base de dados. Através de regras de sintaxe e semântica (WAHL; HOWES; KILLE, 1997; WAHL, 1997; WAHL et al., 1997; BRADNER, 1997) pode-se restringir atributos a tipos bastante específicos de dados e construções rigorosas, seguindo regras de comparação e exclusão mútua. Com isso, as consultas pelo protocolo tornam-se mais consistentes e não-ambíguas (HOWES, 1997; HOWES; SMITH, 1997).

Inclusão explicita e dinâmica de relacionamento. O LDAP, assim como o modelo OSI (YEMINI, 1993), diferentemente do SNMP, permite a declaração explicita de relacionamentos. Estes são recursos significantes para o gerenciamento integrado de dispositivos de rede. Atributos de relacionamento podem ser bastante úteis para o estabelecimento de correlações e dependências entre dados e eventos em geral, como no exemplo apresentado em (YEMINI, 1993).

Flexibilidade, mobilidade e eficiência. A flexibilidade de representação dos dados, tanto estática quanto dinâmica, e eficiência na manipulação de diretórios *online* tornam o LDAP uma tecnologia útil para os mais variados tipos de sistemas. Um exemplo é a integração entre CORBA e LDAP, onde o último é quem armazena as associações entre nomes e referências para objetos utilizados pelo primeiro (ISQUIERDO, 2001). Por fim, permite ainda um certo grau de mobilidade ao sistema.

Todas essas características levaram a escolha do LDAP como protocolo base do `ida`. Mesmo assim, qualquer gerenciador de banco de dados pode ser utilizado, ou desenvolvendo uma interface para gerenciadores de diretórios LDAP como o OpenLDAP(OpenLDAP Foundation, 2004a) ou modificando a interface de comunicação do `ida`.

C.3.2 OpenLDAP

O OpenLDAP (OpenLDAP Foundation, 2004a; ZEILENGA, 2001a) é desenvolvido e mantido pela comunidade de desenvolvimento de programas para o protocolo LDAP (OpenLDAP Foundation, 2004a). Ele é que um conjunto de ferramentas destinadas ao gerenciamento de árvores de informação de diretórios *online* no padrão LDAP (University of Michigan, 1996; The OpenLDAP Project, 2004).

Na versão atual (2.1.30) o OpenLDAP fornece suporte a utilização das bases de dados GDBM (GNU Operating System - Free Software Foundation, 1999) e Berkeley DB (Sleepycat Software, Inc., 2004). Estes são gerenciadores de bases de dados padrão em sistemas Unix.

No entanto, a interface padrão pode ser estendida para suportar praticamente qualquer outro gerenciador de bando de dados.

C.3.3 Ferramentas de Visualização e Gerenciamento

Um diretório com interface LDAP pode ser acessado por uma vasta gamas de ferramentas. Nesta seção são rapidamente apresentadas algumas ferramentas que permitem a visualização e manipulação dos dados de um diretório LDAP. A variedade e funcionalidade das ferramentas é boa e atende praticamente todos os requisitos para a visualização e gerenciamento de servidores desta tecnologia.

C.3.3.1 Ganymede

Ganymede (ABBEY, 2003; ABBEY; MULVANEY, 1998) é um sistema para o gerenciamento de dados que são disponibilizados a rede através de mecanismos padrões de distribuição como o NIS, DNS e LDAP. Esta ferramenta foi desenvolvida com o objetivo de prover controle rígido sobre quais tipos de modificações podem ser realizadas sobre a base de dados que ela gerencia e possibilitar que múltiplos usuários realizem alterações simultaneamente.

Ganymede foi desenvolvido em Java e possibilita uma administração distribuída e remota de servidores de diretórios como o NIS e o LDAP. Todas essas características da ferramenta tornam-a uma opção interessante para o gerenciamento da base de dados do `ida`.

C.3.3.2 OpenLDAP

O próprio OpenLDAP fornece algumas ferramentas para o gerenciamento da base de dados através do LDAP. Entre elas podem ser citadas as de busca, inclusão, remoção e alteração de registros, respectivamente, *ldapsearch*, *ldapadd*, *ldapdelete* e *ldapmodify* (The OpenLDAP Project, 2004).

C.3.3.3 PerLDAP

Esta ferramenta foi desenvolvida para usuários PeopleSoft (PeopleSoft, Inc., 2004) que desejam sincronizar suas bases de dados PeopleSoft com um servidor LDAP (The Mozilla Organization, 1998).

C.3.3.4 Ldaptor

Ldaptor é um conjunto de programas clientes, aplicativos e biblioteca de programação escritos em Python puro (VIRTANEN, 2003b). O Ldaptor fornece comandos de busca, troca de senha, geração de senhas no formato das contas LDAP, geração do arquivo de configuração `dhcp.conf` baseado nas informações de máquinas do LDAP, entre outros (VIRTANEN, 2003b).

C.3.3.5 Web2ldap

É uma plataforma que permite a implementação de características LDAP sofisticadas (STRöDER, 2004). É considerada um canivete suíço para acesso e manipulação de servidores LDAP sem a necessidade de configurar coisa alguma. Esse sistema possui funcionalidades apuradas como a navegação e exibição de referências e dependências de um esquema LDAP (versão 3). Maiores características e recursos disponíveis podem ser encontrados na documentação do sistema (STRöDER, 2004).

C.3.3.6 Luma

Luma é uma ferramenta gráfica para o acesso e gerenciamento das informações armazenadas em de servidores LDAP. Ela foi escrita em Python e suporta as línguas: Inglês, Alemão, Português e Norueguês (DEPPING, 2004).

C.3.3.7 LDAPUserFolder

LDAPUserFolder é uma interface que substitui o mecanismo de autenticação do Zope (Zope Community, 2004), autenticando os usuários contra um diretório LDAP (Zetwork GmbH, 2004).

C.3.3.8 LDAP Configuration Tool

A ferramenta de configuração LDAP é uma aplicação simples que auxilia na reconfiguração do Linux para tirar vantagem de um diretório LDAP no controle de acesso, gerenciamento de contas e autenticação de usuários (Novell, Inc., 2004).

C.3.3.9 Kantan LDAP Search Engine

A máquina de busca LDAP Kantan (*Kantan LDAP Search Engine*) (REINKING, 2003) é uma interface simples e intuitiva para a consulta em servidores de diretórios LDAP. Ela é desenvolvida em Python e roda em servidores Web através de *scripts* CGI (GUELICH; GUNDAVARAM; BIRZNIEKS, 2000).

C.3.3.10 Directory Assistant

O assistente de diretório (*Directory Assistant*) (SESSINK, 2003) é uma aplicação simples para o gerenciamento de livros de endereços LDAP. Esta ferramenta possui apenas os mínimos recursos necessários para o gerenciamento desse tipo de informações dentro de um diretório LDAP.

C.3.3.11 GQ

GQ (DERDELINCKX, 2002) é um cliente LDAP gráfico baseado na linguagem GTK. Possui entre suas características: um navegador LDAP; um navegador de esquemas LDAP; um construtor de protótipos; permite utilizar qualquer número de servidores; permite buscas baseadas em um único argumento ou através de filtros LDAP; manutenção de entradas LDAP.

C.3.3.12 Directory administrator

O administrador de diretório (*Directory administrator*) (AMADOR, 2004) é uma ferramenta avançada de gerenciamento de diretórios LDAP. Ela pode ser utilizada para gerenciar usuários e grupos Unix, informações de catálogos de endereços corporativos, controle de acesso baseado a nível de máquina e roteamento avançado de e-mails.

C.3.3.13 SMBLDAP tools

SMBLDAP-TOOLS (IDEALX Contributions to the Samba project, 2004) é um pacote que contém *scripts* úteis para o gerenciamento de usuários e senhas em um diretório LDAP que estiver sendo utilizado como base de dados para usuários e grupos (em sistemas Unix ou no Samba). Estes *scripts* desempenham funções básicas como adição, remoção e modificação de entradas de usuários e grupos.

C.3.3.14 MigrationTools

As ferramentas de migração (*MigrationTools*) (PADL Software Pty Ltd, 2004) são um conjunto de programas Perl para a migração de usuários, grupos, *aliases*, máquinas, grupos de rede, protocolos, RPCs e serviços de servidores de nomes existentes (arquivos de texto, NIS e NetInfo) para o LDAP.

C.3.3.15 phpLDAPadmin

phpLDAPadmin é um cliente LDAP baseado em tecnologias Web. Ele prove administração fácil e acessível de qualquer lugar para servidores LDAP. Proporciona visualização hierárquica e funções de busca avançadas. É intuitivo para navegar e administrar um servidor LDAP (phpLDAPadmin Supporters, 2004). Possui ainda várias características e funcionalidades básicas e avançadas para gerenciar diretórios LDAP (phpLDAPadmin Supporters, 2004).

C.3.3.16 JXplorer

JXplorer é um navegador de código aberto para diretórios LDAP (BETTS, 2004). Ele é um navegador LDAP padrão, de propósito geral. O JXplorer pode ser utilizado para ler e consultar qualquer diretório LDAP ou X500 (com interface LDAP). Além disso, suporta operações LDAP padrão como adição, remoção, cópia e modificação e operações complexas como a cópia ou remoção de uma inteira sub-árvore do diretório.

C.3.3.17 LDAP Schema Viewer

O visualizador *online* de esquemas LDAP (LDAP Schema Viewer (alan@akbkhome.com, 2002)) é um projeto útil e prático para disponibilizar via Web uma maneira rápida e intuitiva de usuários visualizarem esquemas LDAP (classes, atributos, definições, sintaxes). O sistema funciona como um centralizador colaborativo de esquemas LDAP. Qualquer usuário pode submeter seu esquema LDAP e visualizar os demais, disponíveis na base de dados.

APÊNDICE D RECURSOS IMPLEMENTADOS

D.1 Interface Padrão

A seguir são apresentados os métodos básicos da interface de conexão do `ida`:

new: cria um novo objeto de conexão. Este método também possui uma lista padrão de servidores da base de dados. Caso o usuário não passar nenhum endereço (ou nome) de máquina por parâmetro, um dos servidores da lista padrão será utilizado. Caso a conexão ou busca forem de escrita, o objeto de conexão irá obrigatoriamente se conectar com o primeiro servidor disponível da lista padrão. Por definição, este é um servidor primário da da árvore de informação do diretório da base de dados.

`inmsDataBaseConnect`: Realiza uma conexão simples, sem diretivas de sincronização de acesso.

`inmsDataBaseWriteConnect`: Estabelece que todas as operações de consulta subseqüentes serão síncronas, ou seja, cada entrada (ou ramo da árvore) consultada será bloqueada até a chamada do método de desconexão ou do método de desbloqueio. Para evitar *deadlocks* existe um agente de manutenção que periodicamente verifica a validade dos bloqueios. O tempo de validade é estabelecido pelo administrador do sistema.

`inmsDataBaseReadConnect`: Invoca uma conexão simples. Este método simplesmente irá invocar o método `inmsDataBaseConnect`.

`inmsDataBaseDisconnect`: Caso hajam conexões sincronizadas, libera todas as entradas bloqueadas. Desconecta da base de dados.

`inmsDataBaseReadSearch`: Método de consulta simples, sem sincronização.

`inmsDataBaseUnlockEntry`: Método de desbloqueio manual de uma entrada (ou um conjunto de entradas e sub-entradas) da base de dados.

`inmsDataBaseWriteSearch`: Bloqueia contra escrita de outros as entradas que estão sendo consultadas na base de dados. As entradas bloqueadas serão automaticamente liberadas quando o método `inmsDataBaseDisconnect` for invocado.

inmsDataBaseSearch: Método de consulta simples. Apenas retorna o resultado da consulta do diretório de informações.

inmsDataBaseEntryUpdate: Atualiza uma entrada da base de dados.

inmsDataBaseEntryAdd: Acrescenta uma nova entrada no diretório de informações.

inmsDataBaseEntryDelete: Remove uma entrada da árvore de dados.

inmsDataBaseConnectionObject: Retorna o objeto de conexão a base de dados. Este objeto é disponibilizado pela classe `idaNMOSdataBaseAccessObject`.

inmsDataBaseHost: Retorna o nome do servidor da base de dados ao qual o objeto de conexão mantém um vínculo.

D.2 Componentes Implementados

A seguir são descritas rapidamente as principais funcionalidades e características dos componentes atualmente implementados no `ida`. São citadas a linguagem de desenvolvimento, as funcionalidades básicas e o modo padrão de operação.

A nomenclatura dos componentes segue o seguinte padrão: nome base do componente + nome (no estilo LDAP) da sub-árvore ao qual o módulo pertence. No final, o nome do componente será igual ao seu nome distinto (DN) da sua entrada na base de dados, com os divisores de domínio substituídos pelo caracter '.'. Exemplo:

nome base: *Simulador01*;

pertencerá ao ramo organizacional: *ou=Geral,dc=domínio,dc=br*;

nome distinto na entrada na base de dados: *cn=Simulador01,ou=Geral,dc=domínio,dc=br*;

nome final do componente: *Simulador01.Geral.domínio.br*.

Através desta nomenclatura os agentes de controle, manutenção e serviços irão localizar os executáveis de agentes e módulos. Um agente apenas possui referências a base de dados. Estas referências levarão a entradas únicas. Chegando a uma entrada desse gênero é feita a conversão do DN para o nome final do componente e este irá ser procurado no sistema de arquivos. Quando encontrado será executado ou carregado pelo agente.

D.2.1 Agentes de Controle

D.2.1.1 Agentes de controle geral

Nome: Agente geral de controle.

Linguagem: Perl.

Funcionalidades básicas: executar periodicamente agentes de manutenção.

Modo de operação: Ativação Manual. Quando o administrador da rede instala o `ida` ele terá que definir quais serão os componentes controlados pelo agente de controle e executá-lo. A definição dos módulos de manutenção que ficarão sob o comando desse agente poderá também ser feita dinamicamente.

Descrição sucinta do funcionamento e implementação: executa periodicamente agentes de manutenção da base de dados. Estes agentes têm funções básicas e vitais ao adequado funcionamento de alguns recursos do `ida`.

Os atributos básicos, com exemplo de valores, desse agente são:

```
inmsRunAgent: cn=Agente1,ou=AgentesDeManutencao,dc=domínio,dc=br
inmsDefaultAgentsDirectory: agentesDeManutencao
inmsPollingTime: 20
```

Nenhum dos quatro atributos listados são obrigatórios. Caso nenhum dos atributos estejam definidos, o agente irá ficar sem trabalho, apenas consultando periodicamente sua entrada na base de dados. Por padrão ele usará o valor global de temporização, caso não haja um valor definido em sua entrada. Este valor determina o intervalo de tempo entre uma execução e outra.

Caso o atributo `inmsDefaultAgentsDirectory` esteja definido o código dos agentes será procurado nesse diretório. Caso contrário, a busca será feita no diretório local.

D.2.1.2 Agente auxiliar de controle

Semelhante ao agente de controle geral D.2.1.1. Uma das poucas diferenças é que este agente executa e tem acesso a agentes e módulos de manutenção menos críticos (não essenciais). Uma de suas funções é evitar a sobrecarga do agente geral de controle.

D.2.2 Agentes de Manutenção

D.2.2.1 Ajuste de Referências Cruzadas

Nome: Ajustador de Referências Cruzadas.

Linguagem: Perl.

Funcionalidades básicas: Ajustar as dependências entre as sub-árvores de descrição dos serviços, de módulos de configuração e atualização e de armazenamento dos arquivos de configuração.

Modo de operação: Automática.

Descrição sucinta do funcionamento: Percorre a árvore de serviços e verifica se existem as entradas para módulos de configuração, atualização e armazenamento dos arquivos de configuração gerados pelos módulos de geração. Verifica se existe um atributo indicando

o local de armazenamento nas entradas dos módulos. Caso não exista, ou existem valores distintos, será ajustado o valor no serviço ou no módulo.

D.2.2.2 Conversor de IPs para Nomes de máquinas

Nome: Conversor de IP para Nome.

Linguagem: Perl.

Funcionalidades básicas: Converter os IPs para nomes de máquinas nas entradas dos agentes locais. Estes agentes são identificados pela máquina aonde estão executando (instalado). A conversão é feita de tempos em tempos, quando necessária, para agilizar o trabalho de verificação dos agentes de monitoramento no momento de uma eventual migração de serviços.

Modo de operação: Automática. É executado pelo agente auxiliar de controle.

D.2.2.3 Validação de Dependências

Nome: Verificador de Dependências.

Linguagem: Perl.

Funcionalidades básicas: Verificar a existência das dependências dos serviços cadastrados no sistema.

Modo de operação: Automática.

Descrição sucinta do funcionamento: Percorre a árvore de serviços, identifica as dependências de cada serviço e realiza uma consulta a base de dados para ver se cada dependência existe. Caso uma dependência não exista mais na base de dados o agente emite um alerta ao administrador do sistema.

D.2.2.4 Ajuste de Dependências de Serviços

Nome: Ajustador de Dependências.

Linguagem: Perl.

Funcionalidades básicas: Verificar todas as dependências de cada serviço e registrá-las na entrada base do serviço.

Modo de operação: Automática.

Descrição sucinta do funcionamento: Percorre a árvore de serviços, identifica as dependências de cada serviço (cada serviço pode ter uma sub-árvore de configuração) e registra todas as dependências na entrada principal da sub-árvore. Isso simplifica o trabalho dos verificadores de necessidade de atualização, pois eles terão apenas que consultar as dependências registradas na entrada principal do serviço.

D.2.2.5 Reciclagem de Chaves de Sincronização

Nome: Reciclador de Chaves de Sincronização.

Linguagem: Perl.

Funcionalidades básicas: Verificar todos os registros de sincronização do sistema.

Modo de operação: Automática.

Descrição sucinta do funcionamento: Este componente varre toda a sub-árvore que mantém as chaves de sincronização em busca de entradas que tenham sido esquecidas. Essa verificação é feita comparando o contador global do `ida` com o contador da entrada. Caso a diferença seja maior ou igual a X (a ser definido pelo administrador do sistema) a entrada é considerada um bloqueio esquecido, sendo imediatamente removida. O contador do sistema é baseado em segundos, ou seja, o valor X será o número máximo de segundos que uma entrada pode ficar bloqueada na base de dados.

D.2.2.6 Manutenção de Códigos de Componentes do Sistema

Para cada tipo de componente do sistema existe um agente de manutenção especializado para fazer a manutenção do código do componente na base de dados. Os agentes de manutenção de código atualmente disponíveis são para os seguintes componentes: módulos de geração; módulos de atualização; agentes locais; agentes de serviços; e pacotes de *software*.

Na seqüência é feita uma descrição genérica de um agente de manutenção de código.

Nome: Mantenedor de Código.

Linguagem: Perl.

Funcionalidades básicas: Comparar os códigos de componentes do sistema (no repositório principal) e verificar se os códigos mantidos na base de dados estão ou não atualizados.

Modo de operação: Automática.

Descrição sucinta do funcionamento: Percorre a árvore de códigos de um determinado tipo de componente e realiza a manutenção desta. A comparação de códigos é feita através do uso de chaves MD5. Quando há uma diferença entre a chave armazenada na base de dados (de um determinado componente) e a chave do arquivo fonte, o código da base de dados será atualizado.

D.2.3 Agentes de Monitoramento

Os agentes de monitoramento tem como função básica monitorar serviços e reportar os estados à base de dados. As próximas seções apresentam um caso específico de monitoramento do IPTables que roda no servidor da base de dados e casos gerais de monitoramento de serviços.

D.2.3.1 Monitoramento do IPTables

Este componente tem por objetivo aumentar a segurança de acesso a base de dados. Ele deverá ser executado na máquina que estiver executando o servidor da base de dados do `ida`.

Nome: Monitor de Acesso a Base de Dados.

Linguagem: Perl.

Funcionalidades básicas: Monitorar o número de conexões estabelecidas por cada máquina cliente com o servidor da base de dado.

Modo de operação: Automática.

Descrição sucinta do funcionamento: Este componente é responsável pela análise de todas as tentativas ou conexões com o servidor da base de dados. O IPTables é utilizado como filtro de pacotes. Logo, o componente de monitoramento analisa os registros do IPTables. Cada cliente terá uma entrada na tabela interna do agente e terá suas conexões contabilizadas. Caso um cliente exceda o limite de conexões permitidas dentro de um determinado intervalo de tempo, será gerado um registro de alerta na base de dados. Este registro, por sua vez, será incluído na tabela de acessos negados do IPTables, que será re-gerada pelo módulo de geração das regras deste serviço e este será atualizado pelo módulo de atualização. A ativação do módulo de geração é determinada por um segundo agente de monitoramento, que analisa os registros de alerta reportados pelo monitor de *logs* do IPTables. O esboço da figura D.1 ilustra graficamente o funcionamento básico dos componentes de monitoramento e controle de acesso do IPTables do servidor da base de dados.

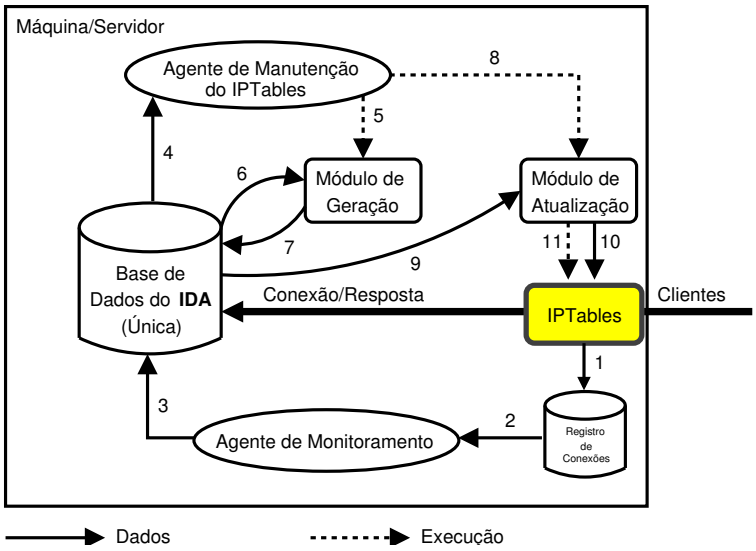

Figura D.1: Visualização do controle local de acesso a base de dados

D.2.3.2 Monitoramento de Serviços

O monitoramento de serviços vitais a rede é importante e imprescindível para de tirar uma radiografia da rede e manter uma boa disponibilidade de recursos. No caso do `ida` esses dados são utilizados pelos agentes de tolerância a falhas. A função destes é avaliar os dados de monitoramento dos diferentes componentes de monitoramento e averiguar qual é o estado de disponibilidade de cada serviço. Caso hajam falhas na prestação de serviços e houverem políticas recuperação ou migração definidas, os agentes de tolerância a falhas entrarão em ação com o objetivo de re-estabelecer a ordem funcional de serviços ou sistemas em geral.

Entre os serviços atualmente com componentes de monitoramento disponíveis estão o DNS e DHCP. Na seqüência é descrita uma visão geral das funcionalidades e aplicação desses recursos.

Nome: Agente de Monitoramento de Serviço.

Linguagem: Perl.

Funcionalidades básicas: Monitorar, analisar e reportar o estado de serviços da rede.

Modo de operação: Automático (independente) ou através de agentes de controle, agentes locais, ou ainda através de um agente de serviço.

Descrição sucinta do funcionamento: Cada agente de monitoramento é responsável pela análise e coleta de dados de resposta de um determinado serviço. Esses dados são convertidos para um formato definido e reportados a base de dados. Na base de dados, serão analisados por agentes que verificarão a necessidade ou não de ações com o intuito de re-estabelecer a disponibilidade de algum serviço que esteja apresentando problemas de funcionamento.

D.2.4 Agentes Locais

Os agentes locais são componentes genéricos, ou seja, no geral existe apenas um único agente local modelo. O que muda de agente local para agente local é a identificação e entrada na base de dados, além das atribuições específicas de cada agente local, como agentes de serviços e módulos sob seu comando.

A figura D.2 ilustra como é o processo de instalação dos agentes locais em cada máquina da rede. Existe uma ferramenta de instalação que precisa ser executada em cada máquina. Esta irá fazer uma cópia do agente local padrão (disponível na base de dados), criar uma nova entrada com uma cópia do agente modelo, registrar e identificar o novo agente local, além de incluir uma nova e única chave de autenticação para o componente em questão. Por fim, basta executar o agente local e associar agentes de serviço e/ou módulos na entrada (da base de dados) de cada agente.

O ID (Identificador Único) do agente local é formado pelo endereço IP e o nome da máquina. Ele é único em domínios aonde não é possível existirem duas máquinas com o mesmo

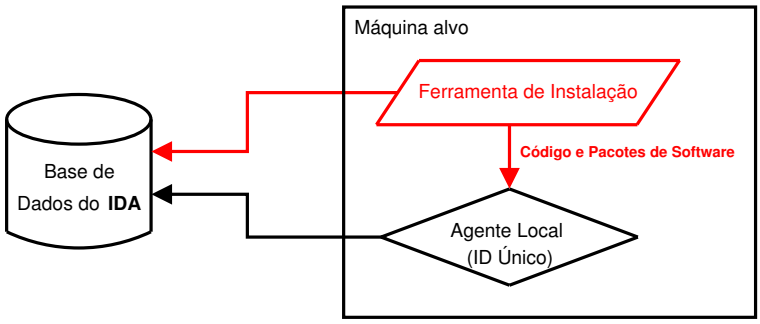

Figura D.2: Agentes locais - instalação

IP e o mesmo nome. Um caso destes é possível de acontecer em domínios com várias sub-redes internas de IPs falsos, locais ou geograficamente distribuídas, interligadas por diferentes roteadores ou simples *gateways* de acesso.

Nome: Agente Local Padrão.

Linguagem: Perl.

Funcionalidades básicas: Controlar, atualizar, ativar e executar agentes de serviços e módulos do sistema.

Modo de operação: Automática, com ativação manual ou programada no sistema local.

Descrição sucinta do funcionamento: Um agente local verifica a lista de agentes de serviço ou módulos sob sua jurisdição e os ativa. Antes de ativar, verifica o estado do código do componente. Caso a base de dados contenha código com chaves de identificação novas, os novos códigos serão baixados para uma posterior execução.

O agente local mantém uma tabela de temporizadores (um para cada componente) para a execução programada de cada agente sob o seu controle. Cada agente local possui um fluxo de execução auxiliar que atende a sinais de execução imediata (estes são mensagens de texto simples que avisam da necessidade de ativação imediata de processos de atualização). Neste caso, os componentes, aos quais os sinais são dirigidos, controlados pelo agente local serão imediatamente ativados. Este último recurso serve para a realização de atualizações urgentes, sendo responsabilidade do administrador do sistema executar manualmente a ferramenta de notificação de execução imediata.

D.2.5 Agentes de Serviços

A maior diferença entre um agente local e um agente de serviço está basicamente na complexidade. Juntar as funções destes dois componentes em um único deixaria este mais complexo.

O agente de serviço, executado e controlado por um agente local, tem um código menor. Pois ele não precisa se preocupar mais com temporizadores de execução, seu próprio código fonte e re-execuções de si próprio.

Nome: Agente de Serviço Padrão.

Linguagem: Perl.

Funcionalidades básicas: Atualizar e executar módulos de geração, de atualização e de execução geral de comandos.

Modo de operação: Automática.

Descrição sucinta do funcionamento e implementação: O agente de serviço verifica, e atualiza quando necessário, os códigos fontes dos módulos sob sua responsabilidade. Outra função é checar a necessidade ou não de execução dos módulos atribuídos a ele. O controle dessa independência entre módulos de geração e atualização no agente de serviço reduz a complexidade e preocupações no desenvolvimento de novos módulos de geração.

D.2.6 Módulos de Geração

A implementação dos módulos de geração depende bastante das necessidades de cada serviço ou máquina. Atualmente estão disponíveis módulos para a geração dos dados de configuração dos serviços de DNS e DHCP. Cada serviço é mantido por um módulo de geração. Este é um componente sem inteligência, porém específico e especializado.

Dando continuidade é apresentado uma descrição geral de um módulo de geração.

Nome: Módulo de geração do serviço X (onde X é um dos serviços anteriormente citados).

Linguagem de desenvolvimento: Perl.

Funcionalidades básicas: Gerar os arquivos de configuração do serviço a partir das informações da base de dados.

Modo de operação: É executado e controlado por um agente de serviço.

Descrição sucinta do funcionamento: o módulo pega as informações de descrição do serviço e os dados relacionados (dependências, heranças) e gera os arquivos de configuração de um determinado serviço. Estes são armazenados na base de dados para posterior utilização pelos módulos de atualização.

D.2.7 Módulos de Atualização

Existem módulos de atualização para os mesmos serviços citados na seção de módulos de geração (seção D.2.6). Estes componentes são bastante simples. No entanto, constituem unidades importantes do sistema para a real manutenção de configurações de serviços e máquinas.

Breve descrição de um módulo de atualização:

Nome: Módulo de atualização do serviço X (onde X é um dos serviços citados na seção D.2.6).

Linguagem de desenvolvimento: Perl, C, Java, *Shell Scripts*, Python.

Funcionalidades básicas: Atualizar os arquivos de configuração do serviço X e re-iniciar o serviço caso necessário (a fim de que as atualizações tenham efeito).

Modo de operação: É executado e controlado por um agente de serviço.

Descrição sucinta do funcionamento: o módulo de atualização simplesmente pega os arquivos de configuração do serviço, os copia para o local apropriado e executa comandos de re-inicialização do serviço caso necessário. Tanto o local de cópia dos arquivos quanto os comandos a serem executados são indicados na entrada (da base de dados) que descreve o módulo. Logo, este pode ser considerado um componente genérico para a atualização de serviços, diferindo apenas nos comandos que deverão ser executados e na localização de entrega dos arquivos de configuração do serviço.

Além dos módulos de serviços também foram implementados módulos de atualização e gerenciamento geral de configurações de uma máquina. A função é ajustar configurações locais como determinar o domínio da rede, a máscara da rede, os servidores de DNS, o *gateway* ou rotas principais e auxiliares, o endereço de *broadcast*, entre outras configurações necessárias para manter um máquina na rede.

D.2.8 Módulos de Execução Geral de Comandos

Um módulo de execução geral de comandos pode ser atribuído a cada agente de serviço. Cada um dos módulos terá uma lista particular de comandos a serem executados, além de uma lista padrão e geral. Esta é útil quando comandos devem ser executados por todos os módulos de execução geral.

A seguir é apresentada uma descrição padrão de um módulo de execução de comandos:

Nome: Módulo padrão de execução de comandos.

Linguagem de desenvolvimento: Perl.

Funcionalidades básicas: Executar listas de comandos.

Modo de operação: É executado e controlado por um agente de serviço.

Descrição sucinta do funcionamento: Estes módulos carregam da base de dados uma lista de comandos (contendo nenhum, um ou mais comandos) e os executam. Após executados, os comandos são assinalados como processados. Quando de uma nova atribuição de comandos para execução os executados são removidos da lista. Essa atribuição de comandos é feita por uma ferramenta específica.

D.3 Ferramentas e Recursos Auxiliares

Ferramentas de manutenção acompanham o sistema mas não são programadas ou executadas automaticamente. Elas servem para o administrador do sistema realizar tarefas como: zerar os contadores de atualização de partes ou de toda a base de dados; editar uma sub-árvore ou entrada específica da base de dados; incrementar os contadores de atualização (chaves que indicam a atualização de um código ou caracterização de um componente ou entrada de dados do sistema); trocar automaticamente todas as senhas de autenticação de todos os componentes e ferramentas do sistema; incluir uma entrada ou sub-árvore na base de dados; apagar elementos da base de dados; apresentar listagens do estado de execução de todos os componentes (distribuídos pela rede) do sistema; carregar entradas de dados (de um arquivo para a base de dados); remover uma árvore de informação da base de dados; testes de sincronização e temporização; entre outras funcionalidades.

As próximas seções descrevem em maiores detalhes algumas ferramentas e recursos disponíveis. Essas ferramentas forma implementadas com diversas linguagens de programação (Perl, Java, C, *Shell Scripts* e Python).

D.3.0.1 Pacote Analisador de Entradas

É um pacote para componentes desenvolvido em Perl e tem como função analisar uma entrada de dados da base de dados. Esta análise consiste na busca e resolução automática de dependências de dados. Com isto é simplificada a tarefa de desenvolvimento de componentes como agentes e módulos de geração, pois estes normalmente necessitam lidar com dependências de dados.

D.3.0.2 Pacotes de Log e Temporização

Estes pacotes registram informações sobre o comportamento da execução dos componentes do sistema. A semelhança entre estes dois pacotes está no fato de ambos reportarem entradas de informação à base de dados. Estes dados podem posteriormente ser visualizados e analisados por ferramentas especializadas.

O **pacote de *log*** permite a instanciação de objetos que encapsulam estruturas para registros de execução de componentes do sistema. Com este pacote qualquer componente do sistema pode facilmente reportar seu estado completo de execução através de diretivas de registro de atividades. O mesmo conteúdo de uma função normal de impressão pode ser passado ao objeto de *log* criado através do uso desse pacote. A seção B.2 do apêndice B apresenta um exemplo de uma entrada de *log*.

A figura D.3 é apresenta um exemplo prático de uso do pacote de *log*.

```
objetoDeLog->acrescentaRegistro(parâmetros);
```

Figura D.3: Exemplo de uso do pacote de *log*

Os `parâmetros` podem ser qualquer conjuntos de cadeias de caracteres e formatações convencionais de impressão.

O **pacote de temporização** funciona semelhantemente ao pacote de *log*. A função do pacote de temporização é fornecer mecanismos que facilitem a cronometragem das mais diversas rotinas, consultas, chamadas de sistemas, atualizações da base de dados, funções ou até trechos de código. Os dados coletados, agrupados e identificados são enviados à árvore de temporização da base de dados. A coleta é feita similarmente a de *log*. Uma das diferenças é que para cada tipo ou coleta de tempo são feitas duas chamadas. Uma que indica o inicio da cronometragem e a outra que indica o final. Ambas recebem um único parâmetro, que é o identificador da coleta. Este identificador pode ser qualquer cadeia de caracteres, incluindo espaçamentos, símbolos e números. A análise e apresentação de dados estatísticos das temporizações armazenadas na base de dados é feita através de uma ferramenta especializada, que permite vários tipos de consultas especialistas e agrupamentos de dados. A seção B.2 do apêndice B apresenta um exemplo de uma entrada de temporização.

Na figura D.4 apresentado um exemplo prático de uso do pacote de temporização.

```
objetoDeTemporização->marcaInício(identificador);
... # código ou rotinas quaisquer de programa
objetoDeTemporização->marcaFim(identificador);
```

Figura D.4: Exemplo de uso do pacote de temporização

O `identificador` pode ser qualquer cadeia de caracteres. O desenvolvedor ou usuário é quem define os identificadores. Por exemplo, um determinado módulo pode conter 4 tipos de consultas à base de dados. Neste caso, uma maneira simples de diferenciar as quatro temporizações seria através dos identificadores: "consulta 1", "consulta 2", "consulta 3"e "consulta 4". Ao final da execução do módulo serão reportados a base de dados os tempos de execução, identificados, de cada uma das quatro consultas.

Tanto os *logs* quanto as temporizações são importantes para acompanhar o estado geral do sistema. Com esses dados em mãos é possível extrair uma radiografia geral do sistema, identificando falhas de vários tipos e problemas de sobrecarga do sistema. Isso possibilita uma rápida identificação e correção de possíveis problemas no sistema. Estes dados são ainda mais importantes e essenciais por se tratar de um sistema de gerenciamento altamente distribuível.

D.3.0.3 Pacotes de Funções Gerais

Os pacotes com funções gerais fornecem ao usuário e desenvolvedor do sistema uma série de métodos e recursos para o desenvolvimento e manutenção dos componentes do sistema. Os métodos providos vão desde validação de caminhos de arquivos até a geração de chaves de atualização únicas. Conversão de nomes de componentes do sistema e de entradas da base de dados, tempo corrente em formato de cadeia de caracteres, IP da máquina local, micro-segundos do instante de tempo atual, nova chave de atualização a partir de uma atualmente em uso, tempo

global de *polling* e modo global de depuração são alguns dos métodos que também fazem parte do grupo de funcionalidades disponíveis nesses pacotes.

D.3.0.4 Pacotes de Funções Específicas

Os pacotes de funções específicas são direcionados a grupos específicos de componentes do sistema. Um tipo próprio de pacotes deste gênero são os que fazem um mapeamento simples entre os nomes reais dos atributos da base de dados e os nomes de atributos utilizados nos componentes do sistema. Alguns destes pacotes são específicos a determinados serviços (módulos de geração em particular). Outros são mais gerais, como os que definem o mapeamento das características de máquinas, pois são utilizados por diversos componentes do sistema.

Outros pacotes de funções específicas incluem pacotes como o de sincronização de acesso as informações do sistema e o de checagem de entradas de máquinas na base de dados. O primeiro é responsável pelo controle de concorrência no acesso de escrita a determinada entrada ou ramo da base de dados. O segundo apenas verifica a existência e coerência de uma determinada entrada que descreve as configurações de uma máquina da rede.

D.3.1 Ferramentas de uso e manutenção

As ferramentas auxiliares do sistema incluem:

editor de entradas: permite a edição de uma entrada ou árvore de informações da base de dados; a edição pode ser feita local ou remotamente; o editor irá estabelecer uma conexão com a base de dados, baixar os dados das entradas, abrir o editor, permitir que o usuário realize alterações e, por fim, as alterações serão enviadas a base de dados;

zerador de chaves: zera os carimbos de ordem e chaves de atualização de um conjunto de entradas da base de dados;

incrementador de chaves: incrementa o valor das chaves de atualização de uma determinada árvore de dados; isso pode incluir incrementos numéricos em chaves formadas por contadores e novos MD5 para chaves de atualização de código;

editor de atributos: permite atribuir valores ou editar atributos individuais de uma entrada da base de dados;

carregador de entradas da base de dados: permite ao usuário baixar e carregar entradas da base de dados; com isso, o administrador pode utilizar ferramentas de edição gráfica ou textual próprias para alterar os dados da base de dados;

modificador geral de domínio: possibilita a troca geral e automática de domínio administrativo; esta ferramenta ajusta as definições de domínio de componentes e da base do sistema; em um único comando o usuário pode definir um novo domínio administrativo (e conseqüente hierarquia da base de dado) para o sistema; além disso, esta ferramenta pode

também ser utilizada para alterar definições (endereços de armazenamento na base de dados) de sub-árvores inteiras;

removedor de sub-árvores de dados: permite, através de um único e simples comando a remoção completa de uma árvore de informações da base de dados;

pesquisador de dados: sua função é prover um comando simples e prático para a realização de consultas corriqueiras a base de dados; fornecendo apenas o endereço base para a consulta o administrador poderá ter como retorno uma única entrada ou uma árvore completa de dados;

D.4 Relatórios de Temporização

Os dois conjuntos de dados apresentados na figura D.5 são parte de um relatório de estatísticas de temporização de componentes do sistema. Eles apresentam basicamente o que foi monitorado (`TAG`), o número de ocorrências e o tempo médio de cada execução. Ao final, é apresentada também uma média geral dos componentes monitorados (`alltogether`).

```
TIMING STATISTICS FOR: DNSprimário.MódulosDeGeração.domínio.br
        TAG: "search 4"     OCCURRENCE TIMES: 1  AVERAGE TIME: 12733
        TAG: "search 2"     OCCURRENCE TIMES: 1  AVERAGE TIME: 3039302
        TAG: "search 1"     OCCURRENCE TIMES: 1  AVERAGE TIME: 32009
        TAG: "search 3"     OCCURRENCE TIMES: 7  AVERAGE TIME: 9053.85
        TAG: "running time" OCCURRENCE TIMES: 1  AVERAGE TIME: 3253570
        TAG: "writing data" OCCURRENCE TIMES: 5  AVERAGE TIME: 338.6
        TAG: "alltogether"  OCCURRENCE TIMES: 16 AVERAGE TIME: 400167.75

TIMING STATISTICS FOR: DHCPprimário.MódulosDeAtualização.domínio.br
        TAG: "search 1"          OCCURRENCE TIMES: 1 AVERAGE TIME: 33528
        TAG: "path verification" OCCURRENCE TIMES: 1 AVERAGE TIME: 88
        TAG: "running time"      OCCURRENCE TIMES: 1 AVERAGE TIME: 40490
        TAG: "writing data"      OCCURRENCE TIMES: 1 AVERAGE TIME: 399
        TAG: "alltogether"       OCCURRENCE TIMES: 4 AVERAGE TIME: 18626.25
```

Figura D.5: Exemplo de um relatório de temporização

Os dois exemplos apresentados foram extraídos da base de dados utilizando a ferramenta *idaTimingReportAnalyser.pl*. Esta permite o agrupamento dos registros de temporização por componentes do sistema, por máquina ou por tipo de registro. Ela possibilita também a consulta por registros referentes a um componente em específico, a um padrão específico (`TAG`), ou a um padrão qualquer de dados. Outra opção é o tipo de unidade de tempo (segundos, mili-segundos ou micro-segundos) e a consulta sobre os registros ainda não processados ou visualizados, o que reduz o campo de busca e a quantidade de dados. No caso da unidade de tempo, o padrão é microsegundos. Os exemplos apresentados seguem a unidade de tempo padrão.

D.5 Relatórios de Atividade

Na figura D.6 é apresentada uma visualização simples de relatórios de *log* no formato descrito na seção B.2.4 do apêndice B. Essa visualização foi extraída com o uso da ferramenta

idaLogReportAnalyser.pl.

```
MODULE: MantenedorDeAgentesLocais.AgentesDeManutenção.domínio.br RUNNING AT
HOST: relho(192.168.179.79)
  [04/01/2005 14:41:24 70 1] => MESSAGE: agentesLocais/gerente.AgentesLocais.
domínio.br.pl module's source file DOES NOT EXISTS!
  [04/01/2005 14:41:24 106 2] => MESSAGE: Config file agentesLocais/gerente.A
gentesLocais.domínio.br.pl ALREADY UP TO DATE!
  [04/01/2005 14:41:24 70 3] => MESSAGE: agentesLocais/Servidor2.AgentesLocai
s.domínio.br.pl module's source file DOES NOT EXISTS!

MODULE: DNSprimário.MódulosDeGeração.domínio.br RUNNING AT HOST: relho(192.16
8.179.79)
  [27/11/2004 19:35:05 27 1] => I'm STARTING TO RUN!
  [27/11/2004 19:35:08 30 2] => I've just FINISHED!
```

Figura D.6: Exemplo de um relatório de atividade

A aplicativo de visualização dos registros de atividade permite consultas baseadas em: máquina de origem dos *logs*, componente do sistema que gerou os registros, período de tempo, tipo de mensagens (exemplo: mensagens de alerta, erro, informação) e através de qualquer seqüência de caracteres ou símbolos que possam identificar diferentes tipos de *log*.

No caso dos *logs* apresentados, eles referem a dois componentes distintos do sistema e são apenas informativos.

D.6 Relatórios de Monitoramento de Sistemas e Serviços

Os relatórios de atividade de serviços são semelhantes os registros de atividade do sistema (seção D.5). O diferencial é que eles representam dados de atividade ou inatividade de máquinas e serviços da rede ao invés de componentes do sistema. A figura D.7 apresentada um exemplo de dados de monitoramento do serviço de DNS.

```
SERVICE: DNSprimário.domínio.br RUNNING AT HOST: relho(192.168.179.79)
  [21/12/2004 22:20:19 24 1] => MACHINE: up and running
  [21/12/2004 22:20:19 45 2] => SERVICE STATUS: working
```

Figura D.7: Exemplo de um relatório de monitoramento

D.7 Exemplo Prático de Código

A seguir é apresentado um código completo de um módulo de atualização do DHCP. Nas linhas 08 e 09 é estabelecida uma conexão com a base de dados. Dando seqüência, dentro da rotina `atualizaArquivosLocais` é feita uma consulta a base de dados (linha 15), buscando todos os arquivos de configuração do serviço DHCP. Neste caso, a profundidade da consulta é todas as entradas abaixo da entrada principal do serviço (identificada pela variável `ENTRADA`). O laço de repetição (linha 19) irá pegar arquivo por arquivo de configuração e gravá-los no diretório de configuração do serviço. Por fim (linha 32) , será executado o comando para efetivação das atualizações realizadas.

```
01 #!/usr/bin/perl
02
03 use strict;
03 use idaPackages::idaDataBaseAccessObject;
04
05 my $ENTRADA = "cn=DHCPprimário,ou=ArquivosDeConfiguração,dc=domínio,dc=br";
06 my $SENHA = "qualquer";
07
08 my $dataBase = idaPackages::idaDataBaseAccessObject->new();
09 $dataBase->inmsDataBaseReadConnect($ENTRADA, $SENHA);
10 atualizaArquivosLocais();
11 $dataBase->inmsDataBaseDisconnect();
12
13 sub atualizaArquivosLocais
14 {
15     my $arquivos = $dataBase->inmsDataBaseSearch($ENTRADA, $PROFUNDIDADE, $FILTROS);
16     my $max = $arquivos->count;
17
18     # baixa e grava arquivos de configuração
19     for ($i = 1; $i < $max; $i++) {
20         my $entry = $arquivos->entry($i);
21         my $arquivoDeConfiguracao = $entry->get_value('inmsDefaultDirectory');
22         $arquivoDeConfiguracao = idaPathVerify($arquivoDeConfiguracao);
23         $arquivoDeConfiguracao .= $entry->get_value('cn');
24         open(DATA,">$arquivoDeConfiguracao") || die("Não pode abrir arquivo");
25         print DATA $entry->get_value('inmsConfFile');
26         close(DATA);
27     }
28
29     # pega e executa o comando de re-inicialização do serviço
30     my $comando = $arquivos->entry(0)->get_value('inmsServiceCommand');
31     if ($comando ne "") {
32         system $comando;
33     }
34
35     return 0;
36 }
```