Universidade do Minho
Escola de Engenharia

Departamento de Informática
Grupo de Comunicações por Computador

CORDENA – Uma Plataforma para Gestão de Redes Baseada em Políticas.

Arquitecturas e mecanismos de tradução de políticas

GUILHERME ANTÓNIO TEIXEIRA

Dissertação apresentada à Universidade do Minho para obtenção do grau de Mestre em Informática na especialidade de Sistemas Distribuídos, Comunicações por Computador e Arquitectura de Computadores, elaborada sob orientação do Professor Doutor Paulo Martins de Carvalho

2005-2006

Dedicatória

Dedico este trabalho aos meus pais pelo valoroso contributo que tiveram criando oportunidades ao longo da vida, e pela forma determinada com que me incentivaram na busca do conhecimento como forma de construção e realização profissional e pessoal.
Aos meus colegas e amigos que me demonstraram sempre que, mesmo tendo as "pestanas queimadas", devemos conseguir ver sempre a amizade, o diálogo e a socialização como forma de evoluir pessoalmente e construir uma personalidade plural.

Mas dedico acima de tudo este trabalho e o desafio que ele representou para mim, à Susana e à Leonor por terem contribuído tanto para a sua conclusão.
Contribuição em apoio incondicional, incentivo e em compreensão pela minha constante presença ausente.

Agradecimentos

Não posso deixar de agradecer também às pessoas ou entidades que de alguma forma, directa ou indirecta contribuíram para este trabalho e para a sua finalização.
Agradeço às empresas do meio profissional onde me integrei no período de tempo que durou o Mestrado de Informática, nomeadamente a Neoplástica e a Sonae Indústria, pelos apoios materiais e de tempo concedido quer para a parte lectiva, quer para a conclusão da dissertação.

Agradeço de uma forma geral aos meus colegas do Mestrado pelo companheirismo e exemplar relacionamento que construímos desde 2001 com esta oportunidade.
Agradeço também a toda a equipa lectiva do Mestrado de Informática, pelo apoio e coordenação de todas as fases de aquisição de novos conhecimentos, em especial ao Prof. Paulo Carvalho pela disponibilidade e apoio na orientação deste trabalho e ao Grupo de Comunicações por Computador do Departamento de Informática pelo acompanhamento e companheirismo aquando da apresentação da Plataforma CORDENA no CRC 2005 em Portalegre.

A todos o meu sincero agradecimento.

Resumo

O alargamento das redes, o aumento da sua complexidade e a heterogeneidade dos equipamentos a elas ligados têm dificultado de uma forma geral a sua gestão e monitorização. Garantir uma determinada qualidade de serviço fim-a-fim, ou disponibilizar de uma forma controlada serviços de valor para suportar processos de negócio críticos, são objectivos complexos a que a área da gestão de redes tem tentado dar resposta.
O presente trabalho propõe uma plataforma para Gestão de Redes Baseada em Políticas – A Plataforma CORDENA e uma forma escalável de transposição de políticas para a rede ao nível do repositório de políticas como forma de minimizar os impactos em termos de desempenho e escalabilidade.
O enfoque principal vai para a modelação mecânica e lógica da arquitectura, para a estrutura do modelo de informação subjacente, e para os mecanismos internos de tradução, responsáveis por transcrever entidades organizacionais em propriedades da infra-estrutura.
São tecidos alguns exemplos de aplicação como forma de elucidar acerca das vantagens no uso de uma implementação com base nos princípios e mecanismos aqui descritos.
É também descrito o ambiente laboratorial virtual com o qual foram testadas diversas tecnologias no decurso dos trabalhos inerentes à investigação associada. Estas incursões técnicas foram incluídas como uma mais valia e uma abordagem interessante para o desenvolvimento, teste e aplicação de políticas em rede que esta dissertação não quis deixar de registar.
É ainda apresentado um conjunto de considerações colhidas no decurso das experimentações efectuadas e no desenho da arquitectura propriamente dita, assim como uma proposta de planeamento.
O trabalho realça ainda alguns trabalhos de investigação, mantendo uma linha de continuidade caracterizada pela criação de redes inteligentes ou auto suficientes, nomeadamente o uso de meta-políticas como complemento ao modelo aqui apresentado.

Palavras-chave: PBNM, PBN, PBM, Gestão de Redes baseada em Políticas, COPS, COPS-PR, LDAP, Políticas, Gestão de Redes, Virtualização, OpenLDAP, Linux

Abstract

The growth of communication networks, their increasing complexity and the heterogeneity of the connected equipments, are making the administration and monitoring tasks difficult to accomplish. Providing a specific Quality of Service from an end-to-end view, or delivering value added services in a controlled way to support critical business processes, are becoming complex goals for which the network management tries to get the right answers.

The present work proposes a framework for Policy Based Network Management - The Framework CORDENA, along with a scalable solution for policy translation from the high business level to the network layer. This is carried out at the repository level in order to minimize the performance impact and to maintain the architecture more scalable.

The main focus is given to the mechanical and logical modeling of the architecture, for the underlying information model structures, and for the internal mechanisms which translate the business objects into infrastructure properties.

In order to understand to what extent the principles and the mechanisms described here can be useful in a real live environment, some application examples are provided.

The test laboratory environment used for the technology research and experimentation is also described since it brings a added value contribution for polices development and testing.

In addition, it is presented an implementation plan and some considerations collected throughout the work in designing and modelling the CORDENA framework.

Finally, this work also highlights some research initiatives that complement the scope regarding the Intelligent Networks research area, namely the meta-policies approach.

Keywords: PBNM, PBN, PBM, Policy Based Network Management, COPS, COPS-PR, LDAP, Policies, Network Management, Virtualization, OpenLDAP, Linux

Índice de Conteúdos

Índice de Figuras

Índice de Tabelas

Acrónimos

ACL	Access Control Lists
API	Application Programming Interface
ASCII	American Standart Code for Information Interchange
CIM	DMTF Common Information Model
CLI	Command Line Interpreter
COPS	Common Open Policy Service
COPS - PR	Common Open Policy Service – Provisioning
CORBA	Common Object Request Broker Architecture
DEN	Directory Enabled Networks
DHCP	Dynamic Host Configuration Protocol
DiffServ	Differentiated Services
DIT	Directory Information Tree
DMI	Desktop Management Information
DMTF	Distributed Management Task Force
DMZ	Demilitarized Zone
DN	Distinguished Name
DNS	Domain Name System
DSCP	Differentiated Services Code Point
DSML	Directory Service Markup Language
EPD	Encoded Provisioning Instance Data
ERP	Enterprise Resource Planning
GPF	General Protection Fault
HTTP	Hypertext Transport Protocol
IETF	Internet Engineering Task Force
INMF	Internet Network Management Framework
IntServ	Integrated Services
IP	Internet Protocol
ISP	Internet Service Porviders
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MIB	Management Information Base
NAT	Network Address Translation
NIS	Network Information System

NOS	Network Operating System
NSIS	New Steps In Signaling
NTP	Network Time Protocol
OSI	Open Systems Interconnection
PAM	Plugabble Authentication Modules
PBM	Policy Based Management
PBN	Policy Based Networking
PC	Policy Console
PCELS	Policy Core Extensions LDAP *Schema*
PCIM	IEFT Policy Core Information Model
PCIME	IETF Policy Core Information Model Extensions
PDB	Per Domain Behavior
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PHB	Per Hop Behavior
PIB	Policy Information Base
PMT	Policy Management Tool
PP	*Policy proxy*
PR	*Policy Repository*
PRID	Complete Provisioning Instance Identifier
QoS	Quality of Service
RDN	Relative Distinguished Name
RFC	Request for Comments
RMON	Remote Monitoring
RSVP	Resource Reservation Protocol
SLA	Service Level Agreement
SLS	Service Level Specification
SNMP	Simple Network Management Protocol
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TMN	Telecommunications Management Network
VMM	Virtual Machine Monitors
VPN	Virtual Private Network
WAN	Wide Area Network
XML	Extensible Markup Language

1 Introdução

A evolução dos sistemas de informação tem trazido consigo inúmeras vantagens que têm potenciado de uma forma espantosa a evolução da espécie Humana.
Paralelamente a todas as vantagens, a maior parte delas relacionadas com o automatismo, a simplificação das nossas tarefas quotidianas, e a capacidade que o Homem tem de gerir e controlar o meio que o rodeia, muitas vezes por aparente contra-senso, há um afastamento destes objectivos genéricos, dada a inevitável complexidade em descrever o Mundo e as relações entre todos os objectos relacionados num determinado domínio.
Assim, e mais uma vez, surge a necessidade de reinventar, de recorrer a reengenharia, de renovar e redefinir, de forma a simplificar e novamente correr no sentido da abstracção, ao mesmo tempo que mantemos todas as vantagens e pressupostos da evolução e do que entretanto foi conseguido pela inovação constante.
A cada dia que passa há assim a necessidade de inventar novas formas de descrever o Mundo, já ele diferente e mais complexo que antes.
Esta breve introdução visa apenas contextualizar a presente dissertação, pois na sua essência é justamente numa destas movimentações que se enquadra.
Navegando pelos próximos Capítulos, o leitor terá a oportunidade de conhecer um Modelo de Gestão de Redes Baseada em Políticas que persegue justamente a automação, optimização e extrapolação da Gestão de redes, ao mesmo que aposta numa abordagem simples, tentando sempre que possível combater a complexidade, sem que esse ponto altere de alguma forma o objectivo nuclear a que se propõe.

1.1 Enquadramento Temático

O alargamento das redes, o aumento da sua complexidade e a heterogeneidade dos equipamentos a elas ligados têm dificultado de uma forma geral a sua gestão e monitorização.
Garantir uma determinada qualidade de serviço fim-a-fim, ou disponibilizar de uma forma controlada serviços de valor para suportar processos de negócio críticos, são objectivos complexos a que a área da gestão de redes tem tentado dar resposta.
Na verdade, o controlo eficaz dos novos serviços disponibilizados pelos fornecedores de serviços, criam novas necessidades também em termos de gestão e manutenção.
Face a esta complexidade, vários modelos de gestão foram surgindo desde o *Telecommunications Management Network* (TMN) iniciado nas redes de telecomunicações, até ao *Internet Network Management Framework* (INMF), entre outros.
A verdade é que a gestão e monitorização da rede, de uma forma ou de outra vai sendo executada, mas evidenciando uma clara falta de integração.
Geralmente os administradores de rede vêem-se obrigados a gerir uma amálgama de equipamentos baseados em diferentes tecnologias, obrigando a controlar vários domínios de conhecimento diferentes, em áreas tão separadas como a gestão de um sistema operativo de rede, ou a configuração de um *router* e sem qualquer tipo de integração.
Assim, o interesse em uniformizar a gestão destes recursos, é claramente importante e permite definir a um nível de abstracção superior, as políticas de gestão da rede e as definições concretas dos serviços pretendidos, ocultando o detalhe técnico não necessário.
Gestão de Redes Baseada em Políticas - *Policy Based Network Management* (PBNM), é uma evolução nos princípios de Gestão de Redes, ao favorecer a visão global dos sistemas, de modo a controlar eficazmente Qualidade de Serviço (QoS) fim-a-fim, acessos via *Virtual Private Networks* (VPN), etc, e de uma forma integrada com todas as políticas guardadas e geridas em entidades centrais.
A definição de políticas na rede a um nível superior de abstracção, além de afastar o detalhe técnico não nuclear, abre simultaneamente a porta a várias vantagens indirectas, nomeadamente aumentos de produtividade, reduções significativas no tempo necessário para implementar e executar, etc.

1.2 Objectivos

Apesar de ainda se encontrar numa fase precoce, esta tecnologia já provou que poderá resolver muitos dos problemas que hoje em dia existem em gestão de redes. No entanto, muito trabalho está ainda em desenvolvimento, nomeadamente na definição de standards e interoperacionalidade entre o que de facto é a realidade hoje.
Independentemente das particularidades e propostas específicas desta dissertação, tal é um dos objectivos nucleares da Gestão de Redes Baseada em Políticas por si só e ao qual este trabalho se acrescenta destacando nas suas diversas componentes as respectivas contribuições científicas descritas na Secção seguinte.
De uma forma geral, a abordagem humanizará inclusivamente a Gestão de Redes, equiparando a Gestão de Infra-estruturas à Gestão de Equipas, de onde se extraem algumas conclusões interessantes aplicadas ao modelo aqui proposto.
No final, pretende-se criar uma linha condutora que integra as contribuições entendidas compatíveis com o modelo proposto, sendo esta uma proposta de arquitectura e modelo de informação para a Gestão de Rede baseada em Políticas com um dicionário fortemente indexado ao negócio, uma das particularidades divulgadas pelos organismos responsáveis por tal tecnologia.

1.3 Contribuição Científica

O trabalho realizado e a presente dissertação apresentam uma proposta de arquitectura global para Gestão de Redes Baseada em Políticas, a "Plataforma CORDENA" e uma forma escalável de transposição de políticas para a rede ao nível do repositório de políticas. O enfoque principal vai para a modelação mecânica e lógica da arquitectura, para a estrutura do modelo de informação subjacente, e para os mecanismos internos de tradução, responsáveis por transcrever entidades organizacionais em propriedades da infra-estrutura.
São tecidos alguns exemplos de aplicação como forma de elucidar acerca das vantagens no uso de uma implementação como base nos princípios e mecanismos aqui descritos. Para o apuro técnico necessário em alguns dos domínios tecnológicos aqui subjacentes foi necessário testar e montar várias entidades integradas ou autónomas em ambiente de rede. Para esse fim foi construído um ambiente laboratorial totalmente virtual com uma boa representatividade das infra-estruturas típicas em qualquer empresa. Uma vez que a sua importância foi por demais evidente no decurso deste trabalho, estas incursões técnicas foram incluídas como uma mais valia e uma abordagem inte-

ressante para o desenvolvimento, teste e aplicação de políticas em rede que esta dissertação não quis deixar de registar.
A propósito de uma possível implementação, passível de ser agendada para trabalho futuro, inclui-se um conjunto de considerações colhidas no decurso das experimentações efectuadas e no desenho da arquitectura propriamente dita, assim como uma proposta de planeamento. O trabalho realça ainda alguns trabalhos de investigação, mantendo uma linha de continuidade caracterizada pela criação de redes inteligentes ou auto suficientes, nomeadamente o uso de meta-políticas como complemento ao modelo aqui apresentado.

1.4 Organização da Dissertação

O documento escrito está organizado em cinco principais Capítulos desde o seu enquadramento, passando pelos conceitos teóricos e técnicos fundamentais, pelos detalhes das contribuições científicas, até às sugestões de trabalho futuro.
Os cinco capítulos indicados são:
1. Introdução: Pretende contextualizar o trabalho, delinear os principais objectivos e salientar as contribuições científicas do trabalho realizado. Expõe a Gestão de Redes baseada em Políticas e a sua importância na inovação e evolução dos Sistemas de Informação de uma forma geral.

2. Gestão de Redes Baseada em Políticas: Detalha o que é a Gestão de Redes baseada em Políticas em todas as suas componentes. Entendido como fundamental, este Capítulo parte das definições basilares das arquitecturas subjacentes, políticas, e restantes objectos constituintes, passando pela análise de cada componente da arquitectura e das suas particularidades.
Finalmente, levanta algumas das aplicações desta abordagem às redes e extrai as respectivas conclusões das matérias descritas.

3. A Plataforma CORDENA: O núcleo da dissertação. Debate as razões que estiveram na génese das abordagens sugeridas na dissertação, os princípios e a fundamentação do modelo descrito.
Detalha o modelo nas componentes do modelo de informação, no fluxo de processos e operações, nos processos internos do Servidor de Políticas e da restante arquitectura. Completa-se o Capítulo com as perspectivas de aplicação com alguns exemplos práticos.

4. Incursões Experimentais: Descreve a implementação prática do laboratório virtual assim como o conjunto de considerações e especificações colhidas no decurso do trabalho e sobre a passagem do modelo à fase de implementação.
Além dos resultados finais e os produtos obtidos, é também incluído o modelo de planeamento do projecto subjacente por se entender ter no seu seio valor acrescentado no planeamento de projectos similares.

5. Conclusões: Descreve as principais conclusões do trabalho efectuado, nas abordagens defendidas, no desenvolvimento efectuado, e nas possíveis aplicações futuras do Sistema apresentado. Contextualiza os resultados alcançados no momento actual das Tecnologias de Informação, no que respeita à Gestão e Controlo de Redes de Comunicação por Computador. São abertas ainda as oportunidades de investigação futuras na mesma linha condutora defendida neste trabalho numa secção aqui incluída.
Enumera algumas funcionalidades que por motivos de âmbito ficaram de fora do presente trabalho, e que poderão constituir matéria para trabalho futuro complementar.

O trabalho completa-se com os tradicionais anexos onde serão acrescentados os documentos de cariz técnico detalhado que por motivos de facilidade de leitura não foram incluídos no corpo textual do documento principal.

2 Gestão de Redes Baseada em Políticas

Este primeiro Capítulo pretende esclarecer as motivações que viabilizaram este novo modelo, assim como uma breve abordagem à arquitectura subjacente com base nos *Request for Comments* (RFC) que deram origem a esta nova tecnologia nomeadamente o RFC 3060 e mais tarde o RFC 3460 do grupo de trabalho *Policy Framework*.

Como veremos, a visão holística do modelo confere-lhe uma funcionalidade invejável a todos os níveis, pois de uma abordagem desintegrada e dificilmente funcional, passamos a poder definir por todo um domínio, políticas que se aplicam a toda a organização, e até estendida a relações inter-domínios.

É na realidade um passo em frente em termos de gestão de tecnologias de informação.

2.1 Introdução

Detalhar-se-á seguidamente esta tecnologia, inicialmente com uma visão geral do que é PBNM e de seguida com o detalhe, funções e características das entidades mais importantes desta estrutura de controlo.
Pretende-se apresentar uma primeira abordagem às principais características deste modelo e não um estudo sistemático e profundo dos detalhes de cada entidade, o que virá a ser analisado noutras secções posteriores mais direccionadas.

2.1.1 O Passado Recente

As primeiras ideias relativamente ao conceito PBNM surgiram em 1990.
No entanto, as expectativas na altura foram objecto de alguma frustração, pela grande dificuldade em utilizar as primeiras soluções encontradas. Em Maio de 1997, as empresas Microsoft® e Cisco® anunciaram o que se viria a chamar – *Directory Enabled Networks (DEN) Iniciative* e que pretendia integrar redes e serviços de directório, com vista a disponibilizar ferramentas centralizadas de gestão e monitorização de redes.
Este modelo foi posteriormente submetido ao *Distributed Management Task Force* (DMTF) [123], em Novembro de 1998, o qual o integrou com o seu modelo *Common Information Model* (CIM), até porque o modelo DEN era baseado no modelo CIM.
Chegamos aos dias de hoje, com duas grandes organizações a trabalhar na definição de normas. Por um lado, o DMTF com o modelo indicado antes e, por outro lado, o *Internet Engineering Task Force* (IETF) [122] mapeando o modelo DEN para o serviço de directório *Lightweight Directory Access Protocol* (LDAP) e definindo protocolos auxiliares de distribuição de políticas e interoperacionalidade entre as entidades da arquitectura.
O trabalho no IETF é organizado em grupos de trabalho (*Working Groups*) de onde se destacam vários trabalhando nesta área, dos quais este trabalho fará várias referências aos grupos de trabalho *Resource Allocation Protocol*, que abordou os trabalhos sobre *Common Open Policy Service* (COPS) para distribuição de políticas, e *Policy Framework* onde se centra o trabalho sobre Políticas.

2.1.2 Os Desafios Actuais

Até agora, a gestão de redes tem sido bastante centralizada nos equipamentos específicos, digamos que unitariamente, actuando directamente sobre estes equipamentos de uma forma desintegrada do resto da estrutura e tirando deles relatórios e estatísticas.

Esta abordagem carece obviamente de integridade com o resto da estrutura, além de inevitavelmente encarecer o preço de manutenção global da estrutura.
A necessidade de pessoal cada vez mais especializado para fazer a manutenção destes equipamentos é também algo a ter em conta.
Existe ainda alguma falta de uniformização em termos protocolares. Enquanto que determinados fabricantes privilegiam o suporte nativo a *Simple Network Management-Protocol* (SNMP), outros ou não o fazem, ou enveredam por soluções proprietárias, incompatíveis com os standards. O saldo final de tudo isto não é de modo algum positivo e obriga quem tem de garantir determinados níveis de serviço, a usar um sem fim de ferramentas e consolas, dependendo é claro da complexidade da rede em causa.
O foco hoje em dia vai claramente para os equipamentos, para as interfaces e não para os utilizadores e serviços, que a rede deve servir e disponibilizar respectivamente.
Esta é uma das grandes diferenças patentes no modelo PBNM, onde o enfoque vai para o segundo grupo de objectos.
Juntamente com os problemas de gestão propriamente ditos, emergentes da complexidade crescente das redes, a evolução dos mercados em termos de comunicações de voz e dados (novos serviços, necessidades de qualidade de serviço e garantias), assim como o aumento brutal da concorrência, têm ilustrado a vantagem que seria ter um modelo de gestão que permitisse gerir tudo isto de uma forma simples e em paralelo com as necessidades do negócio.
A Gestão de Redes Baseada em Políticas veio colmatar estes requisitos sendo por isso bastante aliciante, quer para empresas, quer para fornecedores de serviço.

Na base dos novos serviços disponibilizados, podemos encontrar por exemplo a recente "webização" de muitos processos empresariais inter e intra empresas, acelerando-os e acrescentando-lhes valor. As Intranets e Extranets por exemplo, permitem distribuir informação rapidamente, assim como dotar as organizações de alguma agilidade adicional na reacção às solicitações de mercado.
A gestão empresarial baseada em arquitecturas *Enterprise Resource Planning* (ERP) distribuídas é outro exemplo que não podia deixar de ser citado, pela sua representatividade crescente. Hoje cada vez mais empresas enveredam também por este tipo de abordagem de gestão que garante uma integração funcional completa entre todas as áreas funcionais da empresa. Arquitecturas com dezenas de localizações produtivas, com o seu sistema de informação ERP remoto num único Centro de Dados acessível 24hx7, são também soluções viáveis graças às tecnologias de comunicação existentes.
Todas estas necessidades fazem com que cada vez mais as estruturas tecnológicas de suporte adquiram substancial relevo como inevitável base de suporte a todos estes processos.
Por outro lado, as organizações não têm vulgarmente como seu negócio principal as tecnologias de informação, e pretendem usufruir de todos os serviços tecnológicos

necessários à boa prossecução do negócio, ao menor custo possível, sem comprometer a qualidade das soluções.
Por estas razões, a filosofia da gestão de redes baseada em políticas, assume o papel de uma solução a ter realmente em conta por aliar a tecnologia necessária ao cumprimento do negócio em questão, de uma forma mais funcional que técnica.

2.1.3 O que é PBNM?

A Gestão de Redes Baseada em Políticas PBNM é antes de mais uma mudança na forma como as redes são mantidas hoje em dia. Tal como já mencionado antes, em vez do focar o controlo nos equipamentos, baseia-se principalmente nos utilizadores e aplicações. O sistema baseia-se num conjunto de entidades em rede que dinamicamente fazem associações e mapeiam os utilizadores aos equipamentos e serviços, de acordo com as politicas definidas, "escondendo" da visão do administrador da rede o detalhe técnico destes mapeamentos, nomeadamente as configurações.

Este modelo não é apenas uma substituição do que já existe, mas uma importante extensão aos métodos de gestão existentes, como veremos ao longo deste trabalho. O modelo apresenta algumas vantagens importantes, entre elas destacam-se por exemplo as seguintes:

- Permite ao administrador da rede mapear recursos de rede com necessidades de negócio, definindo perfis de qualidade de serviço garantida para processos críticos de negócio.
- Simplifica, de uma forma geral, a gestão da rede tornando os operadores mais produtivos ao simplificar a gestão e configuração dos equipamentos de rede, assim como a implementação de novos serviços.
- Ao permitir um controlo central relativamente a segurança, prioridades de tráfego, acessos VPN, etc, garante-se que todas estas políticas são aplicadas de uma forma consistente por todo o domínio de gestão, mantendo a rede mais consistente e sob controlo.

De uma forma geral, a configuração dos equipamentos de rede, como por exemplo os *router*s e *switches*, pode ser automatizada, facultando às organizações uma ferramenta indispensável para aplicar QoS, por exemplo.

Este modelo prevê ou incorpora a integração de serviços de directório, de onde pode adquirir conhecimento de utilizadores, postos de trabalhos, impressoras, etc. A automação ao nível do inventário é também outra vantagem bastante importante, quer para o administrador de rede, quer para o gestor de sistemas de informação. Com esta integração, a criação de políticas adquire uma nova dimensão, por ser possível a conjugação

de variáveis muito importantes em termos de negócio. O serviço de directório LDAP é disto um exemplo.

Os administradores de rede deixarão de necessitar de se preocupar mais com a configuração em Cisco® *Internet Operating System* (IOS) de um determinado *router*, mas apenas com a visão global de toda a rede, a um nível de abstracção bem superior.

Os objectivos são nobres, e evidenciam a importância do esforço de normalização, compatibilização e mapeamento entre toda a gama de serviços de informação e equipamentos existentes no mercado. Esforço esse, neste momento, em evolução pela acção das entidades referidas anteriormente, o IETF e o DMTF, juntamente com todas as entidades, empresas, utilizadores, etc.

2.1.4 O que são as Políticas?

Uma política é um conjunto de regras e instruções que determinam o funcionamento da infraestrutura.

As políticas podem definir a visão ou normas de serviço da organização em causa, ao transpor as regras de como a rede corporativa deve ser utilizada pelos empregados, colaboradores, clientes ou visitantes de um *website*.

A abrangência desta definição é enorme e na verdade, dependendo da dimensão, as políticas podem estar agrupadas funcionalmente por áreas mais específicas, ou geridas por vários administradores, cada um com a sua responsabilidade.

Por exemplo, enquanto que um bloco pode dizer respeito às políticas de acesso à Internet, definindo quem e quando tem acesso, outros podem ser assignados à gestão de prioridades por atribuição de diferentes marcas a determinados tipos de tráfego com origem em determinados departamentos e em determinadas datas. Trata-se apenas de um exemplo, mas que ilustra já a integração de informações dos serviços de directório e a simplicidade de gerar resultados com o mínimo de complexidade em configuração.

No caso específico da qualidade de serviço, de notar a potente ferramenta que é a abordagem na gestão e definição de comportamentos ao nível de um nodo (equipamento de rede), ou de um domínio, ao facilitar a definição de uma forma expedita e rápida do comportamento para todo o domínio de gestão.

De notar também que até hoje, muitos tecnologias de infra-estrutura tocaram já de alguma forma a questão da Gestão baseada em Políticas. Veja-se o caso dos serviços DHCP, DNS, as próprias *System Policies* do Windows NT e mais recentemente as *Group Policies* do Windows 2000/2003 a *Active Directory*, etc. Analisando a definição, temos de concordar que se tratam da mesma forma de políticas, pelo seu carácter de controlo central e tipo de definição. Na verdade são-no, mas afastam-se claramente da noção aqui apresentada por não haver qualquer tipo de integração com o resto da estrutura.

PBNM compreende toda a rede, em equipamentos, serviços, aplicações, utilizadores e outros objectos, tudo integrado de uma forma consistente num repositório único. No entanto, alguns destes serviços terão de ser integrados na arquitectura PBNM, ou a arquitectura terá de estabelecer uma comunicação bidireccional com eles, de modo a seguir o percurso da integração total, o que veremos em Capítulos seguintes.

2.1.5 Requisitos para PBNM

Os quatro requisitos básicos que devem ser satisfeitos para uma completa implementação de PBNM [68] são:

- **Um modelo de informação** extensível a toda a estrutura e topologia da organização a tratar. Este modelo deverá incluir, por exemplo, os equipamentos, serviços, aplicações, utilizadores e a forma como se relacionam os objectos entre si.
- **Uma linguagem de especificação** para representação de necessidades de negócio e funções, com alguma abstracção de modo que garanta a total independência das questões e sintaxes específicas dos equipamentos e das soluções proprietárias dos diferentes fornecedores. Esta questão está longe de ser simples e implica métodos de compatibilização entre distintas formas de armazenamento e relacionamento da informação, entre directórios ou bases de dados diferentes.
- **Uma plataforma de gestão completa para administração e controlo**, resolução de conflitos e distribuição, onde se inclui a definição da arquitectura, com um peso importante em toda a escalabilidade e consistência do modelo. Este é provavelmente um requisito crucial para o sucesso do modelo, e nesse sentido muito desenvolvimento está a ser feito com relação a protocolos de comunicação intra domínios e entre as diversas entidades do sistema de políticas baseadas em COPS, *New Steps In Signaling* (NSIS), *Extensible Markup Language* (XML), etc. A comunicação entre domínios PBNM diferentes é também outra funcionalidade importante, nomeadamente no que respeita a toda a interoperacionalidade entre eles, partilha de repositórios, extensão de políticas de segurança e relações de confiança entre eles.
- **Uma forma escalável de transposição ou tradução das especificações** de alto nível independentes dos equipamentos, para configurações dependentes e específicas dos equipamentos. Protocolos como SNMP, COPS, ou até *Command Line Interpreter* (CLI), podem ser uma solução interessante para distribuir as políticas pelos diferentes equipamentos na rede. No entanto, isto implica que cada fabricante implemente funcionalidades de tradução das políticas recebidas, para a semântica e sintaxe específica das linguagens dos seus

equipamentos e sistemas. Este ponto é obviamente de progressão mais lenta e depende do envolvimento de muitas entidades e da adopção do standard por parte delas.

Como veremos no decorrer deste trabalho, os desenvolvimentos patentes nesta dissertação enquadram-se nos dois últimos requisitos identificados aqui, na arquitectura da plataforma geral de gestão e nos modelos de tradução escalável de objectos de negócio para variáveis de rede.

2.1.6 Arquitectura Subjacente

O foco de onde são controladas todas as políticas aplicadas no sistema é, sem dúvida, a *Policy Console* (PC), podendo ser mais do que uma, dependendo da dimensão da rede. A consola é onde o administrador da rede aplica, altera e define todas as políticas, juntamente com a visualização do estado da rede e toda a monitorização possível.

Eventualmente coexistente com a consola, tem-se o *Policy Management Tool* (PMT), que poderá ser apelidado como o motor de raciocínio deste sistema, do qual a consola é a sua face.

Esta ferramenta pode inclusive fazer validações em relação às entradas efectuadas pelo administrador e controlar automaticamente a ocorrência de conflitos entre políticas, quando diferentes políticas tentam aplicar diferentes acções aos mesmos objectos, com condições igualmente válidas.

Sempre que há criação de políticas e alteração de políticas, estas têm obrigatoriamente de existir em alguma parte e serem salvaguardadas de alguma maneira. Para isso, existe o *Policy Repository* (PR) onde as políticas são efectivamente guardadas e de onde são consultadas e actualizadas. Estes repositórios podem ser baseados em bases de dados relacionais, ou em directórios. Ambas as abordagens têm vantagens e desvantagens, como mais adiante teremos oportunidade de detalhar.

Para que todo o esforço de manutenção das políticas seja aplicado, estas necessitam de ser correctamente transpostas para os equipamentos alvo. O *Policy Decision Point* (PDP) tem como função principal aceder ao repositório de políticas e colher as políticas a aplicar. O PDP é onde a tradução dessas políticas é despoletada e eventualmente onde são convertidas para um nível de abstracção inferior. Os motores de compatibilização com a infraestrutura estão aqui. Poderão ainda coexistir sobre este conjunto de processos servidores, outras entidades como por exemplo monitores que recolhem informação de estado da rede para avaliar as condições das políticas que a eles recorrem para moldar o comportamento da infra-estrutura.

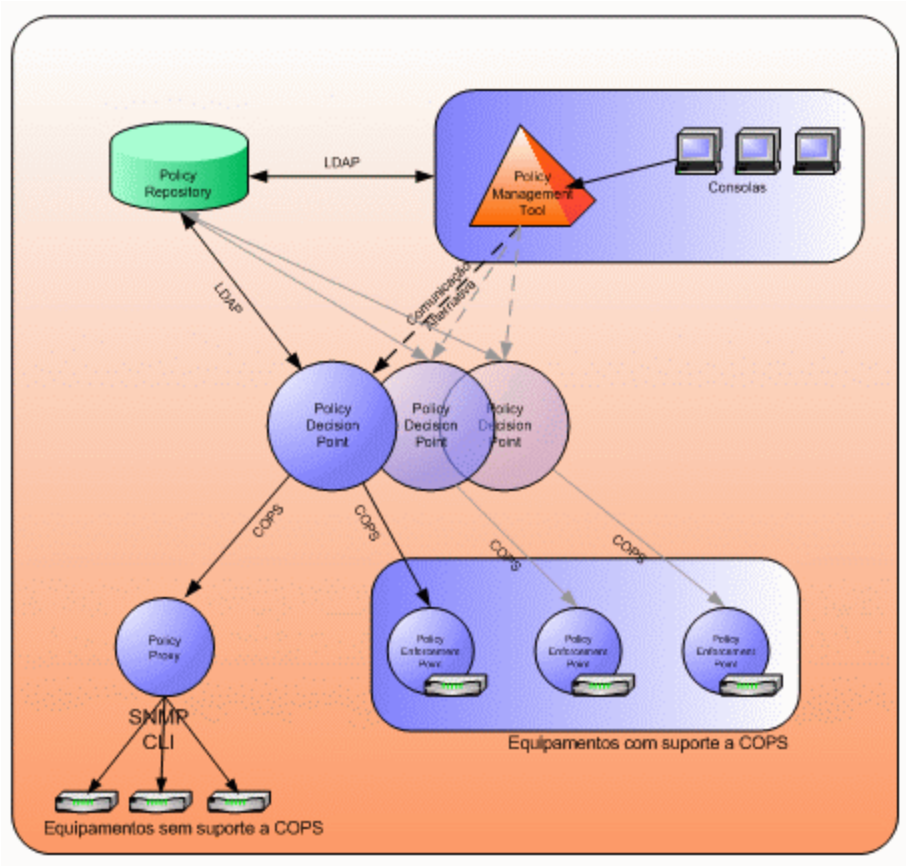


Figura 2.1 – Arquitectura geral para PBNM.

Resumindo, os PDP baseiam as suas decisões nos pedidos de rede, nas políticas do repositório e na monitorização da rede. Vejamos por exemplo a aplicação de uma política que defina uma prioridade de tráfego superior para o pessoal do departamento logístico à sexta-feira, a partir das 16:00h, implica o processamento do pedido, a análise da política respectiva, juntamente com a monitorização do relógio do sistema, sincronizado com um servidor *Network Time Protocol* (NTP) por exemplo.

Finalmente, no final da cadeia aparecem os *Policy Enforcement Points* (PEP), responsáveis por aplicar as políticas nos equipamentos específicos e garantir que são correcta e consistentemente aplicadas nos equipamentos terminais de rede.

Existem ainda os *Policy Proxies* (PP), cuja função visa integrar na rede equipamentos que não estão preparados para interpretar políticas, conseguido a sua integração através do uso de outros protocolos auxiliares de gestão de redes como o SNMP, uma vez que não suportam COPS ou outro protocolo de transporte nativo de políticas.

A Figura 2.1 apresenta o diagrama da arquitectura onde todas as entidades aqui descritas aparecem representadas.

2.1.7 Notas Finais

Está completa esta primeira abordagem às principais definições de PBNM. Os Capítulos seguintes debruçar-se-ão sobre as políticas, arquitectura e as entidades principais deste modelo, com um pouco mais de detalhe.

Em termos de resumo, ficamos a saber quais as principais atribuições desta tecnologia, focada na definição de comportamentos, regras e políticas, a um nível de abstracção superior, e os objectivos nobres que têm vindo a ser perseguidos.

A descrição dos requisitos obrigatórios e da arquitectura, ilustrou também quais os principais problemas em termos de normalização e as grandes mais valias do modelo.

2.2 As Políticas

Neste Capítulo, serão apresentadas as políticas, a sua constituição e algumas breves propriedades importantes. Questões como a extensibilidade por mecanismos de herança, ou os diversos níveis de abstracção são questões interessantes debatidas nesta secção. A importância das políticas em todo este modelo justifica o seu tratamento em separado.

2.2.1 A Estrutura

O bloco mais elementar de uma política é uma regra (*Policy Rule*).

Trata-se de uma simples declaração onde a um objecto é atribuído um valor, como no exemplo da Figura 2.2.

```
Destino = SapServer              // exemplo de um destino.
Prioridade = PremiumService      // exemplo de uma acção.
```

Figura 2.2 – Exemplos de regras.

As regras contêm uma ou mais condições e acções. As condições filtram as situações em que as acções definidas devem ser aplicadas. Veja-se o exemplo da Figura 2.3.

```
If (((trafficToOrFrom AccountingSubnet) and
    (dayOfMonth is last10days))
then
    priority = high
endif
```

Figura 2.3 – Exemplos genéricos de políticas

No exemplo da Figura 2.3, a regra está entre o *if* e o *then*, e a acção entre o *then* e o *endif*. O exemplo apresentado é bastante simples e serve apenas como ilustração. Fica, no entanto, a noção de que as combinações possíveis na construção das condições são bastante extensas, potenciando uma gestão funcionalmente apreciável. Além disso, as regras podem ser combinadas dando origem a políticas, as quais podem ser simples ou complexas. As simples contêm um conjunto de condições e acções, mais ou menos extenso, como ilustrado anteriormente. As complexas, além de incluírem o mesmo que as simples, acrescentam a interacção entre objectos. Exemplos destas políticas podem incluir reconfigurações dinâmicas da rede, baseadas em determinados eventos. As utilizações são infinitas.

As políticas têm ainda outra propriedade importante: a herança, facilitando a construção de políticas mais complexas a partir de outras já construídas e das definições herdadas. A Figura 2.4 ilustra a árvore hierárquica das políticas de acordo com o RFC 3060 [2]

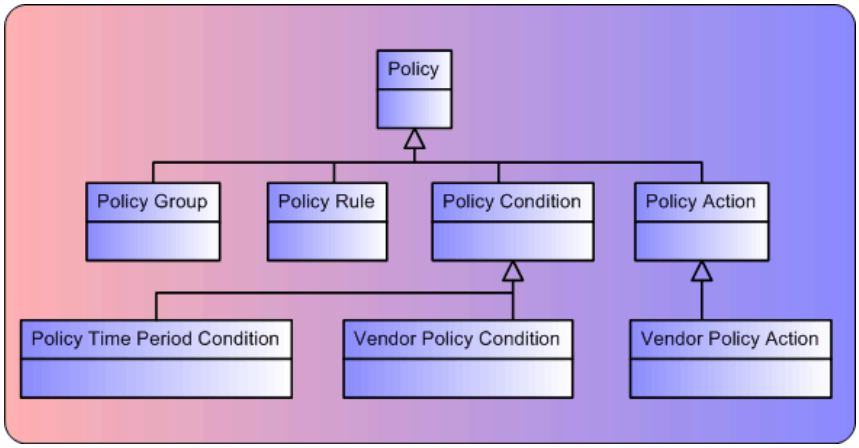


Figura 2.4 – Estrutura hereditária das políticas.

A reutilização e a simplificação da criação são duas fortes vantagens, pela estrutura hierárquica existente. Além disso, a actualização do repositório torna-se bem mais simples. Vejamos o seguinte exemplo da Figura 2.5.

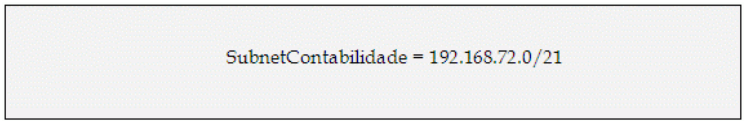


Figura 2.5 – Exemplo de reutilização.

Se por motivos de reorganização, a rede da contabilidade for alterada para outra gama de endereços, a actualização de todas as políticas que lhe fizessem referência seria incontornável. No entanto, desta forma simples, bastará actualizar esta linha, para que todas as políticas que fazem referência à rede Contabilidade usem o valor correcto.

Outra das características interessantes, no que respeita à utilização de políticas, são as *roles*. O conceito de role é bastante importante por permitir agregar grupos de objectos aos quais são aplicadas as políticas, minimizando desta forma o trabalho necessário de aplicação de políticas e aumentando a organização e controlo.

É possível, por exemplo, agregar todas as interfaces de um determinado tipo e aplicar de uma só vez a esse grupo, as políticas pretendidas, em vez de as aplicar repetidamente às várias interfaces. Por outro lado, outras vantagens advêm da sua utilização, nomeadamente a diminuição de algum tráfego em comunicações, uma vez que a mesma política pode ser aplicada a um grupo de componentes.

2.2.2 Tipos de Políticas

Por questões de organização e para facilitar o entendimento das funções atribuídas a cada política, foi estabelecida uma classificação que inclui cinco tipos diferentes.

- **Políticas de Motivação**[1] - Dizem respeito à forma temporal e operacional de como o resultado de uma política deverá ser aplicado. O exemplo citado no RFC 3060 é bastante elucidativo desta questão: Uma tarefa de *backup* poderá estar programado para ser executado entre as 8:00h às 15:00h, e iniciar-se-á mediante a actividade dos discos nesse período. As políticas de configuração e as de utilização são tipos especiais deste primeiro grupo.

    - **Políticas de Configuração** – Definem as configurações por defeito das entidades controláveis na rede.

    - **Políticas de Utilização** – Controlam a selecção e configuração de entidades de rede, baseadas em dados recolhidos de utilização da própria estrutura. Exemplos poderão ser a reconfiguração de um serviço

[1] Não foram encontradas traduções definidas em Português para os tipos de políticas. As traduções indicadas são uma versão pessoal dos termos indicados no RFC 3060.

de *forwarding*, após verificação que um utilizador é membro de um grupo de serviço *Premium*.

- **Políticas de Instalação** – Definem que objectos podem ser instalados num sistema ou componente do sistema e definem também a configuração da forma como devem ser instalados, suas dependências, etc.

- **Políticas de Eventos e Erros** – Definem quais as tarefas a executar, quando ocorrem anomalias no sistema, quais os alertas a gerar, quem deve ser notificado, ou que aplicações executar.

- **Políticas de Segurança** – Controlam os processos de autenticação e auditoria, garantindo ou negando o acesso aos recursos pretendidos.

- **Políticas de Serviços** – As políticas que controlam os serviços são as Políticas de Utilização indicadas anteriormente. As Políticas de Serviços são antes indicadas na descrição e caracterização dos serviços em rede. Por exemplo, todas as interfaces *Wide Area Network* (WAN) no *backbone* deverão usar um determinado tipo de *queuing*.

2.2.3 Os Níveis de Abstracção e sua Representação

O modelo PBNM pressupõe uma independência clara das políticas definidas ao mais alto nível sobre os objectos do directório do domínio. Este aspecto é sem dúvida uma vantagem incontestável no modelo.

No entanto, algo terá que ser feito para mapear todas estas políticas guardadas no repositório central, para as configurações específicas dos variados equipamentos, serviços e aplicações existentes na rede e que implementam as políticas definidas.

Para isso, foram conceptualizados três níveis de abstracção na definição de políticas, analisados de seguida [68].

Políticas definidas ao nível do administrador – Onde são indicadas regras de negócio e as políticas da empresa no que respeita a sistemas de informação. A sua definição é sempre abstracta, totalmente independente de tipos de equipamentos e muito menos das suas particularidades de configuração e manutenção. Estas políticas são definidas pelo administrador através da consola e interpretadas em primeira linha pelo PMT usado. A sua representação no modelo é validada pelos *schemas* usados para bases de

dados ou mapeados para LDAP, de modo a guardar as políticas no PR, de uma forma coerente e normalizada.

Os *schemas* são objecto de trabalho actual no IETF e no DMTF, onde se desenvolvem mapeamentos dos modelos DEN, CIM (*Common Information Model*), para directórios compatíveis LDAP.

Políticas independentes dos equipamentos – As políticas definidas neste nível, diminuíram já o seu nível de abstracção e apresentam o que fazer, mas de uma forma genérica para qualquer equipamento do mesmo tipo. De uma política, deste tipo, podem ser derivadas várias dependentes dos equipamentos. Por exemplo, uma única política independente dos dispositivos, poderá definir vários *Differentiated Services Code Point (*DSCP) para categorizar diferentes tipos de tráfego, sob vários equipamentos.

Estas políticas são processadas no PDP, o qual traduz efectivamente as políticas independentes dos equipamentos para políticas dependentes dos equipamentos de destino.

A sua representação pode passar pelo protocolo SNMP e os respectivos *Management Information Base* (MIB), embora seja preferível o uso do protocolo COPS, juntamente com os PIBs. Esta solução é muito parecida com a primeira, mas mais direccionada para *Policy Based Networking* (PBN), no sentido em que facilita a agregação de objectos, pelo uso dos *roles* indicados anteriormente, diminuindo o tráfego necessário, além de suportarem totalmente a distribuição de políticas.

Políticas dependentes dos equipamentos – Aqui as políticas têm de ser aplicadas de acordo com a especificidade dos equipamentos em causa.

Ainda o exemplo anterior, a política independente dos dispositivos terá de ser aplicada em diversos *router*s, eventualmente com mecanismos diferentes para atingir as mesmas necessidades de qualidade de serviço. Uns poderão usar *weighted fair queuing* com duas filas, enquanto que outros poderão implementar apenas *class based queuing*, mas com três filas. Posto isto, é necessário garantir que as políticas independentes do tipo de equipamentos são correctamente transpostas a este nível. A abstracção neste nível é praticamente nula. As políticas são geradas nos PDPs e são enviadas para os PEPs para a sua aplicação.

Para lá destes níveis, há ainda a necessidade de transpor correctamente as políticas dependentes de equipamento, para as configurações específicas destes dispositivos, muitas vezes sob a forma de listas de acessos, filtros, e outras soluções específicas do tipo de equipamento, sistema operativo, etc.

2.2.4 Notas Finais

As políticas são uma nova forma de operar na gestão das redes. O seu entendimento ao nível estrutural e funcional é crítico para uma boa aplicação da tecnologia. De entre as características mais importantes e que devem obrigatoriamente fazer parte de um qualquer sistema com PBNM, destacam-se a reutilização pela herança e estruturas hierárquicas e a utilização de *roles* de modo a agrupar objectos com as mesmas propriedades ou funcionalmente com funções semelhantes.

A sua representação passa também por diferentes soluções nos três níveis de abstracção analisados. Desde os *schemas* das bases de dados ou directórios no *Policy Repository*, até à utilização de SNMP ou COPS consoante a disponibilidade ou enquadramento da rede, com as respectivas definições em MIBs ou *Policy Information Base* (PIB).

Estas questões serão analisadas com mais alguma informação, no próximo Capítulo sobre a Arquitectura em PBNM.

2.3 Arquitectura e Distribuição

Após o conhecimento do que são as políticas e da visão geral do modelo de gestão de redes baseadas em políticas, analisar-se-ão nesta secção os diversos blocos que constituem a arquitectura deste sistema assim como algum detalhe no que respeita a cada um deles.

Desde o tipo de armazenamento no PR até aos mapeamentos entre níveis de abstracção no PEP, muito mais haveria para tratar. No entanto salientam-se alguns pontos por se apresentarem como mais importantes no enquadramento deste trabalho.

2.3.1 Visão Geral da Arquitectura

No primeiro Capítulo, foi já abreviada a arquitectura subjacente a este modelo de gestão baseada em Políticas. Vamos por um lado recapitular o que foi dito e completar com mais alguma informação o modelo apresentado seguindo uma ordem funcional.

As políticas são definidas e aplicadas através de uma interface com o administrador - a Consola. Uma solução interessante e bastante usada é implementar as consolas sob uma plataforma web, desenvolvida sobre tecnologia multiplataforma (ex. Java) e facilmente escalável para várias consolas simultâneas. Apesar da Consola ser a face do sistema, o motor de raciocínio é sem dúvida o PMT, a camada aplicacional do modelo e que processa toda a informação do sistema.

O PMT deverá validar as políticas, detectar erros e conflitos e ajudar o administrador a criar as políticas da forma mais intuitiva e simples possível, dando a maior ênfase possível à utilização de *roles*, reutilização de políticas e à organização funcional do sistema. As suas principais funções são então a edição e a representação de políticas, a validação e tradução de regras e a resolução global de conflitos.

O PMT comunica com o PR onde são guardadas as políticas actualizadas. Nesta comunicação existem duas importantes possibilidades e que podem reflectir a maior ou menor escalabilidade do sistema.

Além da comunicação com o repositório o PMT, os PDPs têm de ter conhecimento das políticas novas ou alteradas a aplicar no sistema. Para isso, ou os PDPs fazem *pooling* periódico ao repositório verificando dessa forma o que há de novo, ou o PMT informa directamente o PDP acerca das alterações quando elas acorrem.

Enquanto que na primeira solução temos que contabilizar atrasos na propagação das políticas, na segunda surge a necessidade de sincronizar as alterações indicadas aos PDPs e ao repositório simultaneamente.

Como se constata, estas soluções implicam um estudo profundo das implicações em termos de robustez, escalabilidade, redundância, fiabilidade, tempo de resposta, para citar apenas algumas. Esta análise deve ser ponderada tendo em conta a topologia de rede, aplicações, equipamentos e serviços existentes na rede.

A comunicação entre o PMT, PR, e os PDP, depende da tecnologia empregue no sistema PBNM, no entanto, como exemplo poderão ser mensagens LDAP, caso o directório seja compatível com LDAP.

Como já referenciado anteriormente, o PDP tem um papel primordial no modelo, assumindo uma posição central em todo o sistema. É o PDP que traduz as políticas do repositório para serem aplicadas pelo sistema, quer através de *Policy Proxies*, cuja função é estender o domínio PBNM a equipamentos sem suporte nativo a interpretação de políticas, quer directamente aos PEPs, através por exemplo da comunicação baseada em COPS e da manipulação dos respectivos PIBs.

Os equipamentos que não suportam COPS poderão através do *Proxy* usufruir da sua gestão integrada com o restante modelo, através de outros protocolos de gestão como por exemplo o SNMP.

Os PDPs têm como funções principais a localização de regras, a adaptação das políticas aos dispositivos e equipamentos de rede, a verificação de requisitos e a transformação de políticas. Por seu lado, os PEPs têm como função operar de acordo com as políticas especificadas pelas regras, poderão também validar algumas políticas e garantir o envio de relatórios e de estado aos PDPs.

Uma das abordagens deste trabalho, e patente no Capítulo 4 refere-se ao facto de poder capacitar mais estes equipamentos de rede dando-lhes mais autonomia na avaliação das políticas recebidas e agindo localmente.

2.3.2 Distribuição

A arquitectura PBNM pode ser distribuída vulgarmente em duas soluções diferentes: 2-*tier* ou *3-tier*. As soluções são diferenciadas dependendo se o PDP e o PEP coexistem na mesma entidade ou não. Os dois modelos apontam para a concentração na primeira camada do repositório, do PMT e da consola.

A segunda camada inclui o PDP e o PEP da versão 2-*tier*, os quais estão separados cada um em diferentes *tier's* na versão *3-tier*. Esta última tem tido maior aceitação, pelo facto de se poder incluir nela a integração de equipamentos tradicionais, que não suportem o controlo via políticas. Isto é conseguido com a inclusão do *Policy proxy*, tal como indicado anteriormente e ilustrado nas Figuras 2.1 e 2.6.

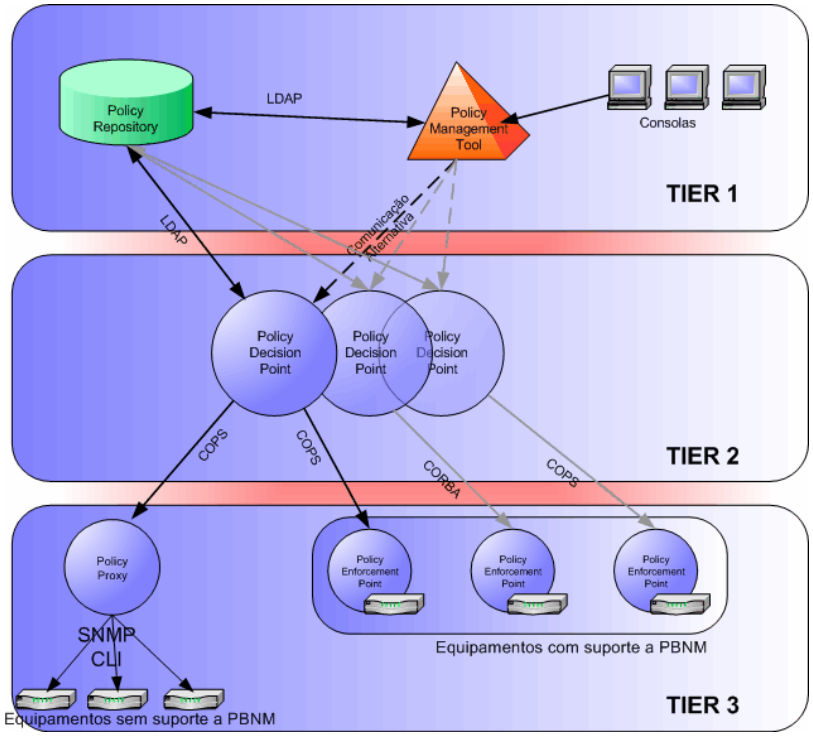


Figura 2.6 – Distribuição em PBNM no modelo *3-tier*.

Outra limitação da versão *2-tier* diz respeito ao número de equipamentos preparados para suportar esta arquitectura, pois o equipamento tem necessidade de assumir as funções de PDP e PEP, o que não existe nos equipamentos actuais. Por outro lado, integrar desde já alguns equipamentos via SNMP ou CLI através dos *proxies* parece uma solução preferível de início. Embora, como é óbvio, existam vantagens em ter um

ambiente homogéneo a operar com equipamentos compatíveis PBNM. No entanto, é bastante difícil que aconteça e as soluções com o *proxy* são uma solução a ter em conta. Vejamos por exemplo algumas vantagens do uso de equipamentos PBNM.

Contrariamente ao que acontece com SNMP onde tem de haver algoritmos de sincronização de conteúdo e de controlo de concorrência, dada a possibilidade de existirem vários gestores para os mesmos agentes. Em PBNM o uso de COPS confere à arquitectura uma leveza importante por associar a cada PEP um e um só PDP, e por essa razão os algoritmos de controlo, quer no PEP, quer no PDP são manifestamente mais simples e eficientes [68], embora haja obviamente capacidade para referenciar PDPs alternativos como medida de protecção contra falhas.

Outra grande vantagem justifica o uso dos *roles*, com os quais pode ser diminuído o tráfego na rede dependendo da arquitectura implementada, uma vez que uma só política pode ser aplicada a um grupo de objectos.

O uso de COPS controla de forma preferível a arquitectura do sistema e a forma como funciona pois através deste protocolo, os PEP têm capacidade para informar os PDPs das suas capacidades e funções, o que com outros dispositivos tradicionais não acontece. Do mesmo modo, os PDPs têm um protocolo nativo para enviar as políticas para os PEPs.

Para finalizar este ponto, convém salientar que as Figuras 2.1 e 2.6 indicam um exemplo na arquitectura PBNM com LDAP. No entanto, outros protocolos e soluções podem ser usados, como por exemplo o *Common Object Request Broker Architecture* (CORBA), assim como diferentes compartimentações das principais entidades da arquitectura nas camadas descritas.

2.3.3 Tolerância a Faltas e Escalabilidade

Uma vez que um dos principais objectivos da PBNM é controlar serviços críticos de rede que suportam processos críticos de negócio, faz todo o sentido replicar e salvaguardar o sistema para evitar problemas de falhas e indisponibilidade. Existem duas entidades que têm obrigatoriamente de ser redundantes; o *Policy Repository* e o PDP. Por outro lado, esta inclusão de redundância, que poderá passar pela replicação do repositório, pode trazer consigo problemas de sincronização.

Para que as diferentes réplicas do repositório não gerem inconsistências por existirem PDP a ler diferentes valores de réplicas diferentes, os PDPs deverão ter uma periodicidade de consulta aos repositórios, inferior à periodicidade com que as réplicas se actualizam. Por outro lado, este pressuposto é variável e depende se o repositório é uma base de dados relacional ou um directório, implicando soluções diferentes e em ambos os casos existindo vantagens e desvantagens que deverão ser ponderadas. Por exemplo, no que diz respeito à escalabilidade, soluções baseadas em bases de dados rela-

cionais podem melhorar a escalabilidade, por permitirem consultas entre objectos relacionados, mais rápidas do que pelas soluções encontradas nos directórios.

Estas consultas são bastante importantes, pois trata-se de uma carga intensa criada pelos PDP na análise das políticas, condições e acções aquando da sua aplicação. Assim, as soluções baseadas em bases de dados têm tido boa aceitação, não só por esta razão, mas pelas restantes características de tolerância a faltas (*commits* e *roll-backs*, *transaction logs*, etc) que neste enquadramento têm uma mais valia importante.

O outro lado da redundância diz respeito aos PDPs onde poderá ser também ponderada a inclusão de novas instâncias com esse propósito. A verdade é que neste modelo, para cada PEP existe um e um só PDP que trata da sua gestão, daí que a inclusão de *backup* para os equipamentos mais críticos seja uma boa solução.

No que diz respeito à escalabilidade dos PDP, cada um poderá controlar um número de PEP que ainda é objecto de estudo. No entanto, em termos muito aproximados, cada PDP poderá controlar na ordem das centenas de PEPs, dependendo da gama dos equipamentos envolvidos [68].

Em estruturas 2-*tier* cada repositório também tem os seus limites, eventualmente muito mais apertados que os PDP indicados antes. De qualquer forma, é claro que a forma de expandir os sistemas com limitações nestas áreas, poderá passar por acrescentar à estrutura mais PDPs em *3-tier*, ou directórios no caso do 2-*tier*.

2.3.4 Notas Finais

Esta secção abordou com algum detalhe adicional, questões relacionadas com o enquadramento deste modelo, das estruturas de rede, quer na problemática da distribuição, quer na tolerância a faltas.

Outro ponto importante, é a definição do tipo de protocolos utilizados e as soluções definidas na distribuição e arquitectura, por poderem condicionar a escalabilidade e funcionalidade da solução geral. Algumas das razões foram também indicadas.

De uma forma geral, não se pretendeu detalhar todas as questões do modelo em termos de arquitectura, mas antes focar alguns pontos importantes juntamente com uma visão relativamente geral do modelo.

2.4 Principais Componentes

Para completar este estudo de PBNM, falta apenas acrescentar algumas notas em relação a cada uma das principais entidades desde modelo indicadas nas Secções anterio-

res. Uma vez que até agora já foram descritas as funções de cada componente no modelo, esta secção indicará algumas características importantes que cada componente deverá satisfazer num sistema PBNM completo e funcional.

2.4.1 Consola e Policy Management Tool

Além do que já foi dito relativamente à consola, nomeadamente acerca das suas funções de *frontend* com o administrador do sistema, salientam-se apenas algumas características importantes neste componente.

- **Facilidade de utilização e um bom suporte gráfico da interface** – embora esta expressão já seja considerada em muitos domínios lugar comum, não deixa de ser um requisito importante e que facilita a utilização das ferramentas.
- **Funcionalidades de *drill-down* e *roll-up*** – uma solução já bastante utilizada, e que provou ser bastante eficaz, é a configuração da interface com recurso a uma árvore de conteúdos. Operações de *drill-down* e *roll-up* sobre esta árvore poderão ser uma ajuda preciosa, se enquadradas com as relações hierárquicas dos objectos representados.
- **Funcionalidades de *auto-discovery*** – Como estas ferramentas não vão à partida trabalhar num "vácuo", as funcionalidades de *auto-discovery* por IP, SNMP, HTTP, etc, facilitam quer o configuração da solução, quer a actualização periódica aquando da ligação de um novo equipamento ou utilizador se assim pretendido.
- **Representação topológica/geográfica** – Fazer um mapeamento, entre o sistema de informação e a representação física dos objectos da organização, é uma forma de por um lado facilitar o entendimento do sistema e a sua transposição para a organização física e por outro lado, facilita também a aplicação das políticas quando as condições levam em linha de conta as variáveis de localização e topologias geográficas.
- **Distribuição de Responsabilidades** – Em grandes organizações é impossível concentrar todas as estruturas sob a alçada de uma única administração. Assim, é imprescindível a distribuição de responsabilidades por tipo de equipamentos, localização geográfica, etc. Para suportar esta distribuição de responsabilidades, a consola e o PMT devem suportar formas de atribuir diferentes gestores a determinados sub domínios.
- **Segurança: Autenticação, Encriptação e Auditoria** – Uma vez que a consola passa a ser um meio privilegiado para guiar alguns processos de negócio na organização, o seu controlo eficaz é crítico. Os aspectos de segurança devem ser muito bem equacionados e não devem menosprezar as três vertentes indicadas no título: autenticação, encriptação e auditoria pelos métodos actuais,

como sejam por exemplo as contas de acesso, *Secure Sockets Layer* (SSL), *logs* do sistema, respectivamente.

- **Inter-operacionalidade entre consolas e outros PMTs** – Útil na interacção entre domínios diferentes e entre áreas de responsabilidade diferentes. Uma utilização da inter operacionalidade entre domínios poderia ser por exemplo, a integração entre empresa – fornecedor de serviço, criando mecanismos através de PBNM, inter-domínios de modo a estabelecer e controlar SLAs entre as duas entidades, de uma forma eficiente e auto suficiente.
- **Facilidade de integração e importação de informação de outras origens** – Uma vez que nestes âmbitos muita informação reside em sistemas de monitorização e controlo empresarial, muitos dados poderão necessitar de ser importados para o PMT. Facilidades de integração destas fontes, criam algumas sinergias importantes no sistema global. Estas integrações poderão passar por exemplo pelo uso de XML ou outras normas de mercado em termos de integração e intercâmbio de dados.
- **Tratamento de conflitos entre políticas** – Existe sempre a possibilidade de serem gerados conflitos com várias condições verificadas como verdadeiras a implicar acções diferentes. Ao nível do PMT, deverão haver funcionalidades adicionais para controlo destas situações. Uma forma de resolver estes conflitos pode ser atribuindo um grau de prioridade a todas as políticas de modo que quando sejam aplicadas, sejam também ordenadas por grau de prioridade. No entanto, ao nível da consola poderá haver alguma filtragem logo à partida.

2.4.2 *Policy Repository*

Os repositórios em PBNM podem ser baseados em bases de dados relacionais ou em directórios (LDAP, por exemplo). Esta escolha pode ter implicações alargadas sobre todo o sistema. De modo a garantir as funcionalidades deste componente há que ponderar sobre os seguintes requisitos:

- **Integridade dos dados** – Este ponto já mencionado anteriormente, diz respeito à redundância e disponibilidade e replicação ou de uma forma geral a tolerância a faltas. Se se tratar de um repositório baseado em serviços de directório podem ser usadas funcionalidades de replicação do directório. Outras soluções podem passar por *Directory Services Markup Language* (DSML), ou um protocolo baseado em documentos XML para troca de informação de directórios entre domínios diferentes. No caso de bases de dados as soluções podem passar pelo uso dos mecanismos de tolerância a faltas comuns aos sistemas gestores de base de dados.

- **Escalabilidade** – De uma forma geral os problemas de escalabilidade que se põem neste modelo de uma ou de outra forma, coexistem também noutros domínios. Neste caso a crescente complexidade das redes, quer em quantidade, quer em variedade, leva-nos também a mensurar na medida do possível as implicações de armazenamento de políticas no repositório, o peso causado pelas mensagens de controlo entre as diversas entidades, a capacidade de processamento necessária para processar todas estas regras e o tempo de resposta às consultas "ramificadas" à base de dados. Por estas razões, o desenho topológico dos repositórios deverá levar todas estas considerações em linha de conta.
- **Segurança** – É crucial que os processos de autenticação sejam funcionais para controlar quem pode efectuar alterações, consultas apenas, etc, inclusivamente com encriptação, dependendo da abrangência do domínio em causa. Como exemplo, HTTP *Digest Authentication* ou SSL são dois protocolos suportados em LDAP que poderão ser usados neste caso.
- **Integração com outras fontes de dados** – Muita informação necessária para a gestão de políticas, pode até nem estar acessível directamente desde o repositório central. Em vez disso, podem por exemplo existir ligações para as fontes externas mantendo da mesma forma a informação integrada. Isto pode fazer sentido, nos casos em que existem dados estatísticos coleccionados via *Remote Monitoring* (RMON), SNMP e que necessitam de ser consultados, assim como por exemplo os *leases* de um servidor DHCP. O tipo de conexão com estas fontes vai depender da latência dos dados (periodicidade das actualizações) e de como o sistema de políticas acede aos dados, podendo até guardar alguma informação de estado localmente, como de uma *cache* se tratasse.
- **Desenvolvimento de *Schemas*** – Apesar de já haver muito trabalho feito relativamente aos *schemas* utilizados, muito há ainda por fazer, embora tenham sido muito recentemente definidos alguns RFCs para *schemas* de base, alguns dos quais fortemente ligados a este trabalho. Uma vez que os *schemas* desenvolvidos no modelo DEN e pelo IETF, podem não servir os propósitos de alguns fabricantes, vão sendo aplicados *schemas* ligeiramente diferentes baseados nos primeiros. Também os administradores dos sistemas, poderão eles próprios achar que o seu repositório carece de informação crítica para o seu caso, procedendo ao seu ajuste. Nesta perspectiva, os directórios são mais flexíveis na alteração da sua estrutura, dando maior suporte a este tipo de alterações que as bases de dados, bastante mais rígidas na sua estrutura.

A grande questão que se põe neste envolvimento é mesmo a diferença entre as duas tecnologias de armazenamento da informação. Por um lado, os directórios são preferíveis para distribuição de blocos de informação e para suportar colecções de objectos

organizados hierarquicamente, o que facilita bastante a gestão de políticas com base na herança. Por outro lado, as bases de dados relacionais garantem um desempenho superior no processamento de consultas complexas aos repositórios e apresentam um conjunto de ferramentas importantes para garantir a integridade das transacções e dos dados.

De uma forma geral os directórios têm-se assumido bem no mercado como repositórios gerais das organizações, também graças aos impulsos de alguns fabricantes, como os casos da Microsoft com a Active Directory e da Novell com o eDirectory para citar apenas dois.

2.4.3 Policy Decision Point

O PDP é uma peça fundamental em todo o modelo PBNM, comunicando por um lado para a camada superior onde se encontra o repositório e o PMT, e por outro para a camada inferior, caso exista uma arquitectura distribuída em *3-tier*, para os PEPs. A forma como esta comunicação se estabelece ainda é algo em estudo com várias possibilidades em aberto. De uma forma geral, Telnet/CLI, CORBA, SNMP, COPS, COPS-PR e HTTP são protocolos possíveis em diversas arquitecturas PBNM.

As questões de fiabilidade e escalabilidade põem-se aqui da mesma forma que foram equacionadas nos restantes componentes do modelo. A forma e o método com estas preocupações são resolvidas, dependem dos protocolos usados. Com COPS, o PEP e o PDP verificam constantemente a sua conexão através de mensagens *keep-alive*. No caso de falha o PEP procura o PDP secundário ou alternativo.

No caso do SNMPCONF, um grupo de trabalho do IETF cuja missão foi entre outras estender o uso de SNMP com extensões para PBNM, a solução passa pela redundância conseguida à custa de várias estações de gestão com os mesmos direitos de acesso aos equipamentos, como indicado no RFC 4011.

Em termos de escalabilidade, a inclusão da versão distribuída *3-tier*, tal como apresentada na secção anterior, resolve já o problema da escalabilidade numa abordagem genérica. No entanto, poderá haver problemas de escalabilidade no processamento interno no PDP. Nos casos em que um número extenso de políticas tem de ser comparado para detectar eventuais conflitos, ou nos casos em que um PDP controla demasiados PEPs, os atrasos podem ser significativos e não desprezáveis, limitando a expansibilidade do sistema.

Finalmente, apenas como resumo indicam-se alguns requisitos a ter em conta neste componente do sistema:

- Suporte para múltiplos protocolos de configuração, como por exemplo CLI, telnet, HTTP, etc

- Redundância, com a inclusão de PDP secundários para os mesmos PEPs.
- Revisão dos procedimentos de tradução de políticas, para detecção de conflitos ou erros.
- Lançamento de alertas associados com conflitos de políticas.
- Segurança

2.4.4 Policy Enforcement Point

Os PEPs são essencialmente os equipamentos terminais ligados em rede, os quais podem ser ou não compatíveis com a aplicação de políticas, integrando-se na arquitectura PBNM.

Assim, exemplos de PEPs poderão ser: *routers e switches, web switches, Traffic shapers, firewalls, remote access servers*, *gateways vpn,* servidores, postos de trabalho, etc. A maior parte destes equipamentos dispensam apresentação. Cada um deles deverá suportar algum protocolo de suporte a PBN, como por exemplo COPS ou COPS-PR, para interagir com o PDP.

A parte aplicacional de suporte a COPS não é muito extensa, podendo ser aplicada através de pequenas actualizações de *firmware* em alguns casos. Caso contrário, estes equipamentos terão de interagir através de um *Policy proxy*.

Pouco mais há a referir em relação aos PEP, até porque ao falar de PDP, também se retrataram os PEPs. Os requisitos deste componente incluem; (i) capacidades adicionais de integração, de modo a integrar facilmente um domínio PBNM de acordo com as normas existentes; (ii) um bom desempenho apesar da quantidade de informação que tem de ser processada; (iii) encriptação e classificação de tráfego, para os casos em que o tráfego está encriptado e o PEP não vê o seu conteúdo.

2.4.5 Notas Finais

Este Capítulo alertou para algumas questões importantes nesta fase inicial em que se encontra esta tecnologia. Foram também enumerados e exemplificados alguns requisitos para cada um dos componentes deste modelo.

No final deste Capítulo será possível deter uma imagem geral do que é PBNM, assim como abrir caminho a algum debate em algumas questões mais específicas dentro do modelo.

Finalmente, as próximas secções resumirão algumas conclusões interessantes acerca do modelo PBNM, assim como algumas aplicações de PBNM.

2.5 Aplicações e perspectivas

Será esta tecnologia compatível com as necessidades de *service providers*? Fará algum sentido a sua aplicação num domínio puramente empresarial em ramos de actividade que não as redes de comunicações e infra-estruturas tecnológicas, onde tais recursos são utilitários perante o *core business* das empresas?

São questões pertinentes e importantes para as quais se tenta ensaiar uma resposta no decurso das próximas páginas.

Passando para o plano mais específico, a gestão de redes baseada em políticas, tem como possíveis domínios de grande aplicabilidade actual, a qualidade de serviço e a gestão de segurança.

No primeiro caso lidando com o controlo de tráfego, quer por reserva com *Resource Reservation Protocol* (RSVP) quer por marcação de pacotes (DSCP), e no segundo auxiliando na configuração de VPNs e no controlo de acessos.

Salientam-se seguidamente, algumas das aplicações de PBNM, encaradas como casos específicos da já mencionada aplicabilidade geral desta tecnologia perspectivando ainda o que poderá ser a sua divulgação geral nos próximos anos.

2.5.1 Qualidade de Serviço (QoS)

À medida que as estruturas de rede, assim como o seu licenciamento, consomem mais recursos financeiros, maior é também a necessidade de encontrar novas formas de rentabilização dos investimentos efectuados.

Como nos últimos tempos a economia tem sido pouco benevolente com as novas tecnologias, surgem novas tentativas aqui e ali para explorar pequenos nichos de mercado, ou soluções pontuais com algumas perspectivas de mercado, mas não as suficientes para uma clara evolução positiva do meio envolvente.

QoS, poderá ser uma clara oportunidade para devolver alguma estabilidade ao meio, por permitir oferecer gamas de serviço personalizadas à medida do cliente e do seu negócio. A junção das novas tecnologias de comunicação aliadas a formas de gestão eficientes, definidas com base no negócio e que simplificam razoavelmente a gestão das redes, parece fazer prever um ciclo positivo no futuro próximo. PBNM é sem dúvida uma ferramenta relevante na implementação de QoS sob a rede.

Na verdade apesar dos modelos de Serviços Integrados (IntServ) e Serviços Diferenciados (DiffServ) serem uma referência, a complexidade das configurações a aplicar nos equipamentos de todo um domínio para a sua utilização, não é desprezável, e poderá contribuir para um relaxamento na aplicação generalizada de QoS às redes hoje exis-

tentes. PBNM fornece um grau de simplificação útil sob todos os aspectos, e acima de tudo contribui para diminuir esta dificuldade.

Outra grande vantagem na aplicação de políticas em QoS é a integridade e uniformização com que é possível aplicar critérios e definições por todo um domínio de gestão. Além disso, outras funcionalidades são acrescidas com o modelo QoS aplicado com PBNM.

No modelo IntServ, dá-se a alocação prévia de recursos ao longo do caminho mais curto entre o destino e a origem. Estas alocações são geridas pelo protocolo RSVP, através de mensagens específicas distribuídas a todos os nós do percurso.

Atendendo a que cada nó do percurso, é aqui um PEP, a decisão de atender ao pedido de alocação de recursos, pode ser redireccionado para o PDP responsável pela sua gestão.

Este serviço, designado de *outsourcing*, pode ser implementado com um protocolo de gestão de políticas como por exemplo o COPS referido anteriormente. Este é um exemplo da organização a que se pode chegar na gestão de QoS numa infra-estrutura.

Por outro lado, existe total integração das políticas com as mensagens RSVP nos modelos propostos até agora.

No modelo IntServ, no processo de reserva de recursos, os pedidos são passados pelos módulos de controlo de admissão e de autenticação, onde várias informações, para lá das contidas numa mensagem RSVP podem ser integradas. Estas informações podem conter, credenciais de utilizador, ou outras informações do directório passíveis de poderem ser utilizadas para proceder a uma correcta autenticação [68]. Os dados contidos nas mensagens RSVP, fornecem também informação de grande valor para PBNM, possibilitando o controlo dos fluxos de onde e para onde vão.

Enquanto que para o modelo IntServ, o protocolo COPS serve os propósitos, para DiffServ, a escolha não é tão simples. Os protocolos para distribuição de políticas em DiffServ, poderão passar por COPS-PR (COPS com extensões para *provisioning*), telnet/CLI, CORBA e SNMP.

No caso de COPS-PR por exemplo, há a salientar que, de uma forma geral, contrariamente ao que acontece com COPS, onde os PEPs recolhem pro-activamente as políticas no PDPs respectivos, com COPS–PR, as políticas são passadas para os PEPs apenas quando há novas políticas definidas a um nível superior, ou após um reinício do próprio PEP, ou pedido do próprio [68].

DiffServ e IntServ têm sido dois modelos de duas escolas distintas, ambos com potencialidades e preocupações. No RFC 2998 propõe-se como alternativa a congregação dos dois modelos simultaneamente, para controlo via DSCP no interior de um domínio, e via RSVP para os *edge router*s.

Aqui, mais do que nos modelos isoladamente, PBNM é ainda mais importante, pois neste caso a configuração de uma solução tal assume níveis realmente elevados de complexidade sendo um alvo interessante para o uso de PBNM nas áreas da integração e controlo transversais.

2.5.2 Segurança

Talvez uma das grandes vantagens na aplicação de PBNM a sistemas de segurança diz respeito mais uma vez a questões de integração.

Temos por um lado o controlo de acessos baseado em *Access Control Lists* (ACL), não nível de rede, e cuja gestão obriga sempre a algum trabalho de configuração detalhado em cada um dos equipamentos.

Por outro lado temos os sistemas operativos de rede que controlam o acesso através de directórios e de forma partilhada, mas apenas para impressoras, pastas partilhadas, *servers* e outros recursos de rede que não *router*s por exemplo.

A gestão de redes baseada em políticas, vem fazer a ponte entre estes dois domínios ao integrar o controlo de acessos baseado na rede e no sistema operativo de rede. Poder-se-á dizer que PBNM permite mapear utilizadores para endereços IP, unificando a estrutura, e abrindo caminho a interessantes vantagens neste âmbito, como facilmente se poderá imaginar. Estas informações podem ainda ser integradas em *Firewalls, Remote Access Servers*, VPN, sempre controladas de uma forma centralizada através do PMT.

2.5.3 Perspectivas de aplicação prática

Se a abordagem PBNM permite gerir de uma forma integrada redes de comunicações pelo "click de um botão", certamente que grande margem de dúvida serão os operadores de telecomunicações os grandes beneficiados pela tecnologia pela simples razão de que esse é seu negócio, o seu *core business.*

Mas não será esta uma perspectiva demasiado redutora face às expectativas criadas em volta da integração com todos os objectos de um repositório LDAP dentro de uma organização? Objectos que são pertença do cliente final e não dos seus fornecedores de serviço.

E se assim é, faz ou não sentido apostar então na extensão da tecnologia ao Mundo empresarial independentemente do ramo de actividade das empresas alvo, baseando a sua aplicação por princípio na vantagem de gerir "tudo" de uma forma integrada?

Entende-se que faz sentido mas dependendo da forma com é implementada!

Certamente que terá grande valor acrescentado, se for dada a devida atenção a características como:

- Transparência na instalação e exploração do *middleware* necessário.
- Total integração com as tecnologias existentes
- Ambiciosas funcionalidades aplicacionais e de *frontend*, com mecanismos amigáveis capazes por exemplo de automatizar e distribuir políticas via "macros" para necessidades e problemas habituais, etc.

Estes três pontos destacam a necessidade de explorar as vantagens da tecnologia mais do que para um óbvio negócio de telecomunicações, onde a tecnologia se justifica por si só.

2.5.4 Notas Finais

Muito mais haveria para retratar neste Capítulo, uma vez que os temas tratados são acima de tudo um começo, e não um domínio acabado do conhecimento.

Qualidade de serviço, é uma tecnologia solidificada a que PBNM trás efectivas vantagens que não devem ser menosprezadas. Da mesma forma, distribuir uma política de segurança empresarial baseada em PBN é certamente mais eficaz, por permitir uma conformidade total ao longo de todo o domínio de aplicação. Além disso, as vantagens de simplificação de configuração são realmente um factor a ter em conta, não só para as configurações previstas, mas também na integração de modelos complementares na busca de soluções cada vez melhores.

Para lá destes exemplos técnicos, vamos ver se o futuro próximo nos facilita a razão pelas perspectivas apontadas no que respeita à aplicabilidade geral dentro do mundo empresarial, justificando-se aqui tal como no plano científico.

3 A Plataforma CORDENA

Tal como mencionado ao longo do presente trabalho, a Gestão de Redes com base em Políticas, abre um novo caminho na gestão das tecnologias de informação, ao permitir transcrever acções e decisões sobre as estruturas tecnológicas de uma forma “inteligente” recorrendo à invocação de objectos, regras, e acções, existentes num serviço de directório que reflecte a organização real em que insere.

Desde forma, é possível através do recurso a linguagem de negócio, implementar um *driver* que definirá e configurará o comportamento da infra-estrutura lógica e física que suporta o dito negócio.

O modelo CORDENA baseia-se num modelo de informação baseado em normas actuais e reflecte uma proposta de tradução da linguagem de alto nível para linguagem de rede, logo ao nível do repositório como forma de ultrapassar problemas de desempenho posteriores.

Integra ainda uma breve abordagem "personificando" a tecnologia ao comparar entidades técnicas de gestão de redes com mecanismos de gestão de equipas, aplicando-os à gestão de redes aqui tratada, ponderando questões como o tipo de controlo e o nível de delegação entre as entidades PDP e PEP.

A designação CORDENA deriva do objectivo principal da plataforma, a coordenação das infra-estruturas. Faz um claro incentivo à acção e a concretização através do seu tempo verbal. Finalmente invoca um dos objectivos cruciais deste trabalho, a simplicidade, traduzida numa palavra comum com o significado pretendido.

3.1 Introdução

Os próximos dois pontos iniciam a descrição do modelo proposto enquadrando-o, definindo desde já onde se situa e qual o âmbito da modelação apresentada.

3.1.1 Enquadramento

O Capítulo 2 forneceu uma perspectiva sobre o que é a arquitectura PBNM, quais as suas potencialidades assim como oportunidades de melhoria.

Face a essas oportunidades, foram surgindo ultimamente várias abordagens para a optimização deste ou daquele aspecto, desde a tolerância a falhas, segurança, organização das políticas nos repositórios, até a extensões aos protocolos de gestão de políticas.

O trabalho apresentado pretende antes de mais estabelecer dentro das várias abordagens sugeridas, uma linha de continuidade caracterizada pela criação de redes inteligentes (*PnP Networks*), ou auto suficientes. Dentro desta linha condutora, e após alguma consulta ao recente trabalho de investigação nesta área, encontram-se várias contribuições de valor que se entende referenciar nesta análise por se enquadrarem claramente dentro desta filosofia de gestão de redes, nomeadamente os trabalhos desenvolvidos sobre meta-políticas. [80] Entende-se desta forma potenciar a investigação sobre uma linha coerente, seguindo o trabalho já desenvolvido, ao invés de um recomeço sobre as mesmas matérias.

Para lá deste breve enquadramento, é detalhada a principal contribuição científica do trabalho, nomeadamente os modelos de informação usados para a especificação do modelo, assim como os mecanismos de tradução de alto nível recomendados para o mapeamento da tecnologia aos objectos de negócio.

São ainda apresentados alguns exemplos de aplicação para melhorar determinadas funcionalidades dentro das temáticas da gestão de redes.

3.1.2 Âmbito de modelação

Para a prossecução dos objectivos traçados, na secção anterior, existem várias partes constituintes já enunciadas no Capítulo 2 e que merecem ser aqui enquadradas no trabalho em curso.

Digamos que de uma forma genérica, os pontos da arquitectura onde reside a maior "inteligência" do modelo serão:

- **O Repositório de Políticas:** Onde já estão guardados todos os objectos e entidades de toda a estrutura da empresa (utilizadores, aplicações, hardware, redes, etc), e onde vamos agora guardar as políticas que iremos usar para gerir a rede.
- **O Servidor de Políticas:** Onde tradicionalmente se materializam os mecanismos de tradução e aplicação das políticas à rede, e onde podem ser adaptadas camadas adicionais para por exemplo monitorizar variáveis da rede, aplicar políticas via *proxies*, entre outras funcionalidades.
- **Os equipamentos terminais:** Onde se espera que as políticas sejam aplicadas de forma autónoma mas recorrendo à ajuda do servidor de políticas para a concretização de condições de rede ou caso necessário, para gestão de conflitos.

Dado o âmbito bastante alargado desta temática e da definição de um tal quadro para gestão de redes, o trabalho presente concentrar-se-á na especificação dos mecanismos de tradução e na clarificação dos modelos de informação necessários para tais mecanismos.

A área destinada à especificação dos processos inerentes aos equipamentos terminais não vai por isso ser objecto de análise em detalhe, embora figure em todo o trabalho a caracterização desse ambiente nas suas mais importantes vertentes, nomeadamente na autonomização de tais elementos, seguindo as linhas orientadoras da secção anterior.

3.2 Os Modelos de Informação

O Modelo de Informação desenhado nesta plataforma levou em linha de conta algumas particularidades relacionadas com a representatividade de uma empresa e não apenas considerações técnicas. O próximo ponto fará um breve enquadramento desses requisitos.

No que respeita ao Modelo de Informação em si, serão enunciados as principais entidades e o seu papel organizacional e/ou técnico. Os detalhes relativos a cada uma dessas entidades será detalhado na secção sobre a arquitectura propriamente dita, pois as definições dos Modelos de Informação estão intrinsecamente ligadas aos mecanismos de tradução de alto nível para a rede descritos adiante.

3.2.1 A Organização de Base

A primeira definição importante foi sem dúvida a organização empresarial de base que serviu de laboratório lógico ao presente desenvolvimento.

De forma a conseguir percorrer os objectivos expostos nas contribuições científicas, e ao mesmo tempo abrir a possibilidade de implementar soluções parciais sob o âmbito de trabalho para alguns dos exemplos apresentados na Capítulo 2, entendeu-se que seria imprescindível ter um ambiente de teste completo com as diversas componentes infra-estruturais que veremos mais adiante, mas também e principalmente a existência de uma organização empresarial fictícia com os seus tradicionais departamentos, localizações, utilizadores, etc, de forma a adequar a plataforma à real envolvente de uma empresa, com a qual se compatibilizou o trabalho.

Assim, sob a actual envolvente foi crucial garantir os seguintes requisitos / características de tal organização:

- Representativa da grande parte das empresas em termos organizacionais.
- Integrando grupos de utilizadores com perfis de acesso diferenciados, nomeadamente grupos departamentais com perfil operacional (logística, produção, etc), e também de administração e controlo.
- Ser uma organização descentralizada por várias localizações geograficamente dispersas.
- Usar sistemas de informação para gerir os seus negócios, quer os operacionais relacionados com o seu negócio central (por exemplo produção), quer também para a administração e controlo de gestão.
- Com uma ligação à Internet para consulta de informação, acesso a serviços de *Internet Banking*, e para gerir e aprovisionar *stocks* junto dos seus fornecedores.
- A empresa disponibiliza também um ambiente de *e-business* para gerir a relação comercial com alguns dos seus parceiros, nomeadamente clientes e distribuidores.
- A organização é uma empresa jovem e dinâmica apostando nas tecnologias e sistemas de informação como factor de diferenciação para criar vantagens competitivas no mercado em que opera. Foram estas as principais características da organização de base entendidas como importantes.

Justamente por se entender que tal definição deveria conter um nível de abstracção tal que permitisse equacionar a aplicação deste modelo a várias topologias de empresas distintas, a definição não contempla outras dimensões de detalhe, que de outra forma deveriam percorrer algumas das dimensões indicadas de seguida. Vejamos algumas

dessas dimensões a equacionar nas soluções encontradas e na forma como são implementadas em produção:

- Volume de negócios
- Capitais próprios
- Mercados alvo
- N.º concreto de colaboradores
- Previsões de evolução
- Tecnologia disponível nos mercados em que opera.
- *Know-how* interno e nível de serviços contratados.
- Estratégia

Muitas mais haveria para citar e analisar, no entanto tal matéria está fora do âmbito desta dissertação. Concretizando finalmente, foi definida uma empresa fictícia com uma distribuição de acordo com a seguinte organização:

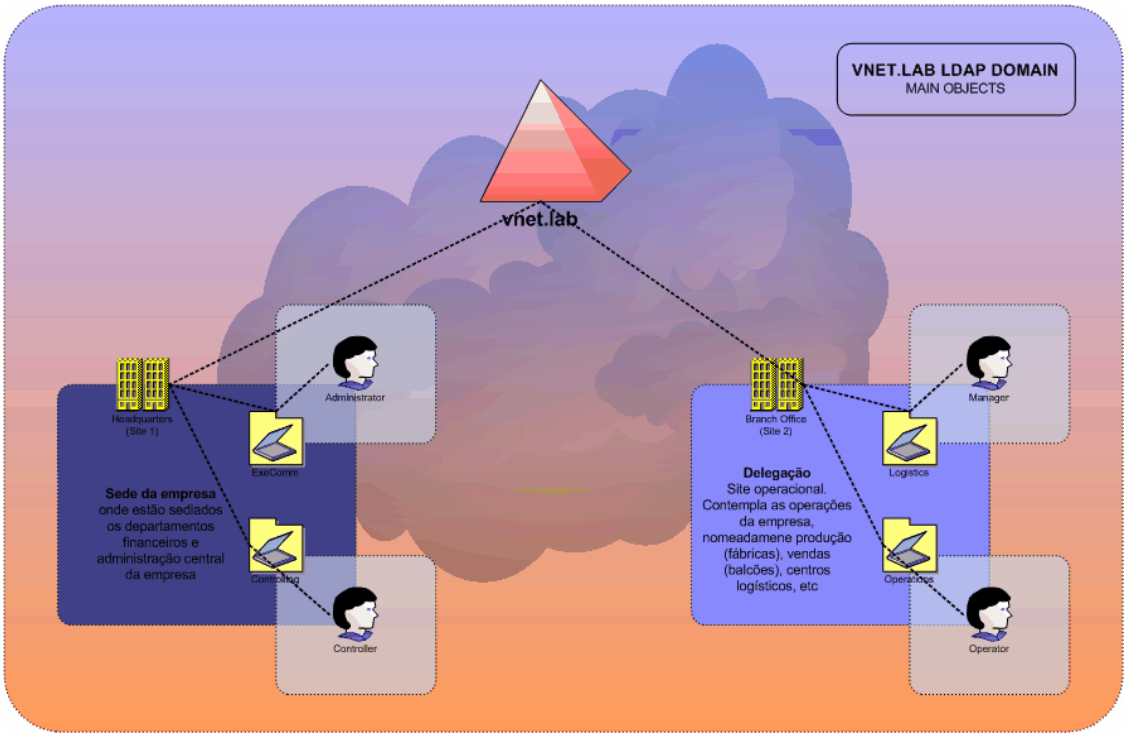


Figura 3.1 – Organização vnet.lab

A organização Vnet.Lab detém um site como sendo a sua sede onde residem habitualmente utilizadores com perfis administrativos nas diversas áreas funcionais da empresa. Para lá da sede, existem várias localizações distribuídas geograficamente onde são executadas as operações de negócio, logísticas, de vendas e distribuição ou até de produção. Entende-se que de uma forma ou de outra, com as devidas variantes, esta estru-

tura é representativa de uma grande parte do tecido empresarial. A Figura 3.1 esquematiza esta topologia organizacional.

A Figura 3.2 ilustra a organização de pessoas e departamentos da organização fictícia que será mapeada sob o modelo proposto em organigrama.

Funcionalmente, poderia ter mais ou menos departamentos, entende-se no entanto que os indicados são os nucleares em qualquer organização e representam os principais níveis de criticidade e importância no seio das empresas.

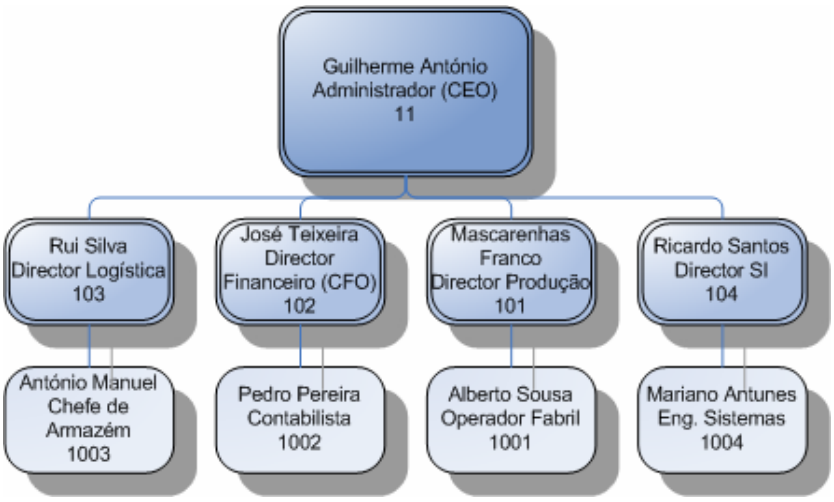


Figura 3.2 – Organigrama da empresa

A empresa está distribuída entre três localizações, sendo uma a sede, duas delegações operacionais. Poderá facilmente ter duas ou mais delegações, sendo as restantes instâncias do modelo actual.

Caso a empresa cresça com mais delegações, as adicionais serão sempre uma cópia das existentes no que respeita ao âmbito do presente trabalho.

3.2.2 Entidades principais

Há a considerar dois grandes grupos de informação. Por um lado, toda a classe de objectos organizacionais que descrevem "aquilo que há" na organização, desde utilizadores, localizações, equipamentos, etc, e por outro as políticas propriamente ditas, sendo que as políticas usarão em larga escala os objectos organizacionais como variáveis a quantificar na criação das suas regras constituintes.

Centrou-se o modelo num conjunto de objectos considerados como os mais importantes e sobre os quais o negócio incide.

Os objectos organizacionais circunscritos ao modelo de informação proposto serão os seguintes:

Objectos	**Exemplos**
Utilizador	António, Manuel, Operador, etc...
Departamentos ou Grupos de utilizadores	Departamento Financeiro, Operadores, etc...
Localizações físicas	Site x, Edifício A, etc...
Ambientes aplicacionais	ERP, Módulo Financeiro, Módulo Vendas, etc...
Processos de negócio	Venda, compra, despacho, etc...
Referenciação temporal	Data/Hora Duração
Variáveis dos Equipamentos Terminais	Interfaces específicos Outras variáveis de estado locais.
Políticas	Regras que aplicam acções sobre a rede aos subconjuntos seleccionados pela condição inicial.

Tabela 3.1 – Objectos organizacionais a tratar

Este é conjunto de objectos identificado que se entende ser suficiente dentro do âmbito deste trabalho para exemplificar os mecanismos de tradução de políticas.

O objectivo principal deste processo é justamente transformar as entidades identificadas na Tabela 3.1, em variáveis e objectos de configuração real na rede, obviamente de uma forma leve que permita encontrar um mínimo peso em termos de recursos de rede e processamento necessários, escalando melhor e funcionando com o máximo desempenho possível. Desta forma poderemos usar tais variáveis na construção das condições de uma forma simples e fiável.

Para lá destes objectos organizacionais a que este trabalho dará destaque pelo enfoque no negócio, outros poderão existir de carácter mais ou menos técnico e que mais não

serão do que mais referências a testar nas condições necessárias aquando da construção de regras constituintes das políticas.

Passando para o outro lado, o lado das acções, será de igual forma importante definir o que se pode fazer com o modelo. Nesse campo, e uma vez que se trata de um trabalho de investigação com aplicação por exemplo na gestão dinâmica de tráfego diferenciado com base em regras de negócio, o trabalho basear-se-á nas capacidades indicadas no RFC 3460 onde é revisto o modelo de informação de politicas agora com algumas extensões importantes e onde se expande o modelo com as entidades necessárias para a implementação das acções relativas à marcação simples de pacotes.

Relativamente às políticas propriamente ditas há a considerar a forma como elas são guardadas e reutilizadas nomeadamente através dos repositórios de políticas internos.

Alguma da informação da plataforma reside em partes distintas da arquitectura. Se todos os dados relativos aos utilizadores residem obrigatoriamente no repositório, as referências temporais podem ser colhidas de um servidor NTP, e o estado das interfaces WAN de um conjunto de *router*s pode por exemplo ser colhido por consulta aos *Management Information Base* (MIBs) respectivos.

Convém salientar que quanto mais extensível for o modelo, maior será a sua aplicabilidade. Por esse motivo lancemos o atrevido desafio de implementar algo que permita o uso de todos os objectos presentes no directório da empresa, pois como sabemos o directório pode ter mais ou menos objectos e ser mais ou menos extensível de acordo com os *schemas* que implementa e os objectos que guarda, e que amanhã podem naturalmente ser mais do que hoje. Talvez o desafio nem seja tão grande quanto isso, pois as variáveis e valores patentes nas regras usadas pelas condições e acções das políticas, podem conter atributos DN o que permite referenciar qualquer objecto do Directório, desde que o seja de forma controlada e com o necessário sentido.

Desta forma estamos a capacitar a plataforma para evoluir na mesma medida que toda a estrutura evolui, o que embora pareça à partida utópico, pode ser possível, obrigando necessariamente a um enquadramento mais abstracto e provavelmente ao ajuste de alguns *schemas* conforme o caso. De qualquer forma como base, o conjunto de objectos indicados é suficiente para testar o modelo.

3.2.3 O Directório

Depois de pesadas as vantagens e desvantagens acerca do uso de directórios comparativamente com as bases de dados relacionais, o modelo foi desenhado sob um directório LDAPv3 face às vantagens que apresenta em termos de rapidez de acesso e pesquisa, e pela facilidade de integração com outros sistemas de directório actualmente usados em grande escala.

Não confundamos com a forma como os dados são fisicamente guardados no servidor, o que pode ocorrer baseado num modelo relacional ou não. Com efeito na implementação CORDENA, o directório usado está fisicamente assente sobre uma instância de uma base de dados não relacional embebida directamente na implementação permitindo assim atingir níveis de desempenho maiores por não serem necessários processos e mecanismos intra-aplicação para registo e consulta de dados.

O passo seguinte foi definir que *schemas* iriam ser necessários para guardar correctamente todas as informações necessárias e povoar o directório com algumas informações para teste. Por isso, uma peça fulcral em todo o modelo proposto, é o sistema de directório usado, neste caso o LDAP, pois sobre ele assenta o Modelo de Informação desenvolvido.

Neste sentido justifica-se enunciar em alguns parágrafos do que se trata e como funciona um directório LDAP na perspectiva do uso das políticas, e dos métodos usados para o armazenamento das políticas, das suas partes constituintes e da sua utilização, de forma a melhor entender as secções posteriores.

3.2.4 A Tecnologia LDAP

Por vezes a necessidade de centralizar toda a informação que circula num determinado domínio, só por si, pode constituir uma vantagem tremenda em domínios onde é difícil a gestão da dispersão, por exemplo com informação dispersa por muitas entidades diferentes, com processos de controlo distintos e com diferentes níveis de qualidade na informação que guardam.

Embora esta perspectiva se refira ao problema do controlo de informação duma forma abstracta, é fácil constatar vários exemplos onde isto acontece. Nós próprios muitas vezes corremos atrás de soluções que nos resolvam o eterno problema de centralizar as nossas informações pessoais, contactos, contas bancárias, cartões, etc.

Se no nosso caso o problema já se verifica, ou pelos menos (para os mais organizados) se reconhece a vantagem que é ter todas estas informações na "palma da mão" sob uma base acessível, simples e flexível, muito mais se verifica numa infra-estrutura de sistemas, onde podem existir um emaranhado de tecnologias diferentes com necessidades de informação distintas e descentralizadas.

No caso mais simples da criação de um novo utilizador na empresa, pode haver necessidade de criar esse utilizador na rede, abrir uma conta de correio electrónico, criar o seu directório pessoal, criar as suas respectivas contas em várias aplicações diferentes, entre outras operações. Cada vez que o utilizador necessita de uma modificação é necessário sincronizar todos os subsistemas para que a informação fique consistente, e ainda fazer o trabalho inverso quando o utilizador deixa a empresa.

A necessidade premente de juntar todas estas informações num único repositório com uma estrutura adequada, está na génese dos serviços de directório. Deter todas as informações relativas aos utilizadores, postos de trabalho, aplicações, etc, sobre uma mesma base é quase uma utopia para o administrador de rede e representa um potencial enorme em termos de produtividade, contrariamente ao que em condições consideradas normais tem de existir para gerir vários sistemas distintos, criar contas, replicar informação, etc.

LDAP é um directório que se tem afirmado como uma norma para este tipo de aplicações. Trata-se de uma norma com origem no protocolo X.500, bem aceite pelos principais vendedores de soluções no mercado e pela comunidade em geral.

O antigo X.500 implementava toda a pilha protocolar OSI, comparativamente ao LDAP que implementa o modelo apenas sobre TCP/IP usando mensagens com um "custo" inferior. A Figura 3.3 mostra a diferença entre uma e outra abordagem.

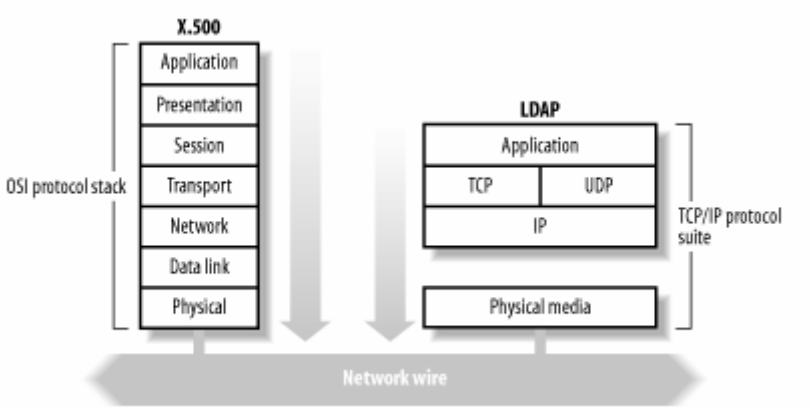


Figura 3.3 – Pilha protocolar X.500 versus LDAP [64]

Além disso, LDAP omite várias instruções pouco usadas no X.500, razão pela qual é chamado *Lightweight* [64]. As principais definições do protocolo LDAP estão patentes do RFC 2251 ao RFC 2256 e em mais uma série extensa de outros RFC que detalham outras funcionalidades e características nomeadamente RFC 2829 para os métodos de autenticação, RFC 3377 para a especificação técnica e o RFC 2830 para as extensões *Transport Layer Security* (TLS) entre outros.

Sem entrar na especificidade do protocolo LDAP de referir apenas que se trata de um protocolo baseado em mensagens cliente/servidor detalhado no RFC 2251 e normalmente implementado em modo assíncrono, o que quer dizer que um cliente pode enviar vários pedidos a um servidor e recebê-los por uma ordem diferente da inicial, tal como ilustrado no Figura 3.4.

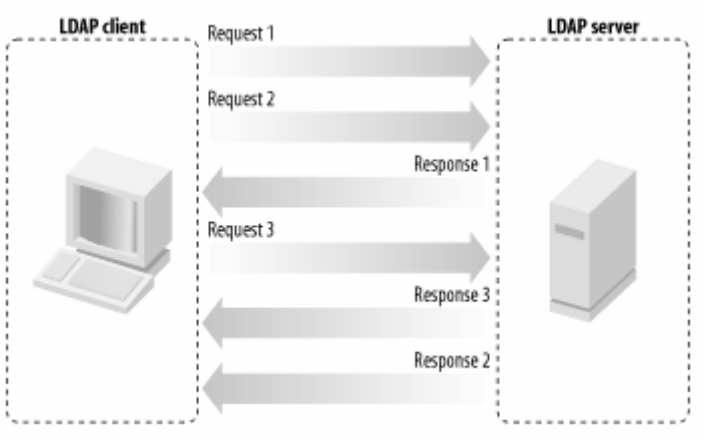


Figura 3.4 – Mensagens LDAP (pedidos e respostas) [64]

Um modelo de informação LDAP inclui a estrutura dos tipos de dados necessários para a construção do directório, preenchido com entradas ou registos que respeitam esta estrutura de dados. Cada entrada descrita tem de conter uma ou mais instâncias de uma classe ObjectClass e os respectivos atributos, uns opcionais, outros obrigatórios, todos definidos nos chamados *schemas*, ficheiros que definem este modelo de informação.

Um servidor LDAP pode implementar paralelamente vários *schemas* simultâneos respondendo assim a vários pedidos de diferentes informações e com diferentes estruturas.

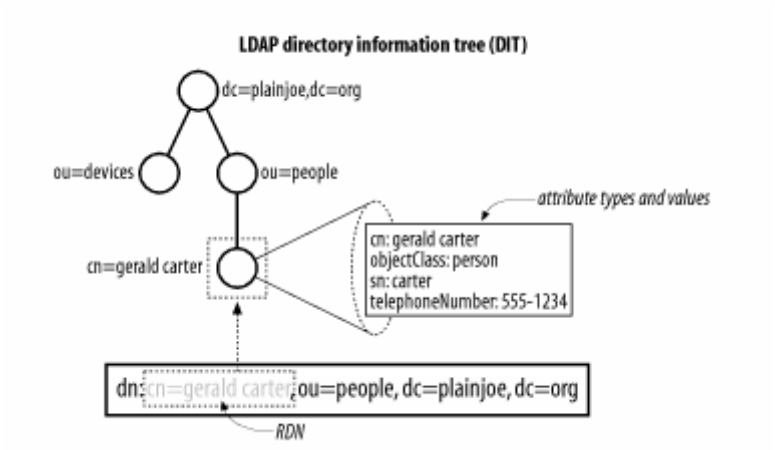


Figura 3.5 – Exemplo de uma árvore de directório LDAP [64]

Sobre esta estrutura de dados, há a considerar um modelo de referenciação e nomenclatura, o qual define a forma como são referênciados os objectos do directório. Cada um tem um atributo que o referencia no ramo do directório em que se situa e que se chama *Relative Distinguished Name* (RDN). Juntando todos os RDN pela hierarquia

correcta, temos o chamado *Distinguished Name* (DN) que identifica o objecto de forma unívoca em todo o directório, tal como indicado na Figura 3.5.

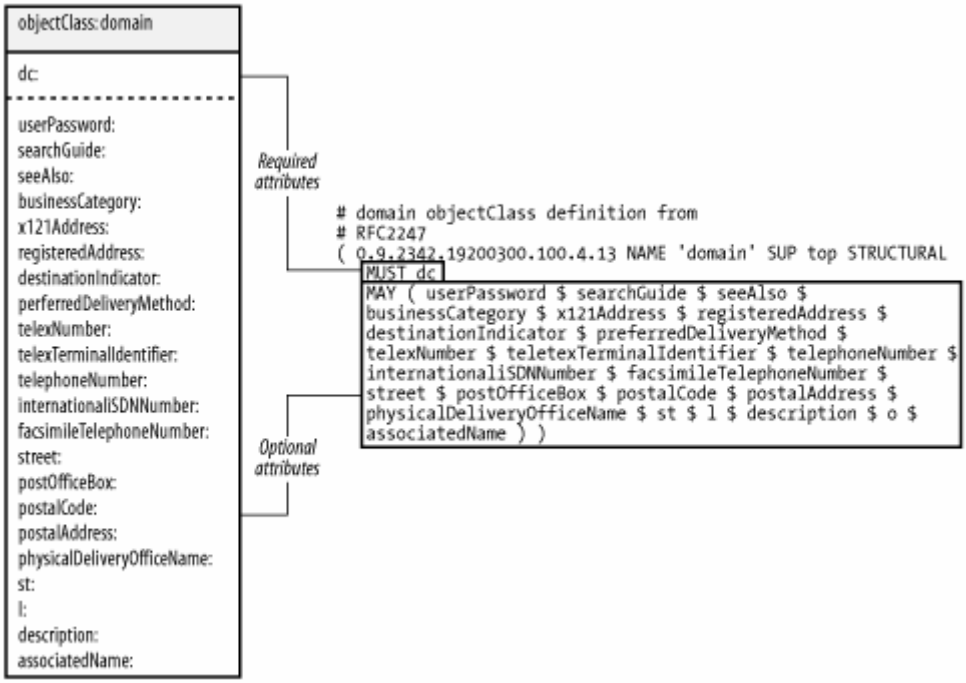


Figura 3.6 – Classe objectClass [64]

A Figura 3.6 apresenta a classe objectClass umas das mais importantes do modelo LDAP, onde se apresenta a distinção entre os atributos obrigatórios (ex. dc) e os facultativos.

3.2.5 A Representação das Políticas em LDAP

De uma forma simples as políticas são constituídas por regras também elas compostas por condições e acções. Estas condições e acções são implementadas com base em pares compostos por variáveis e valores, tal como indicado na Figura 3.7.

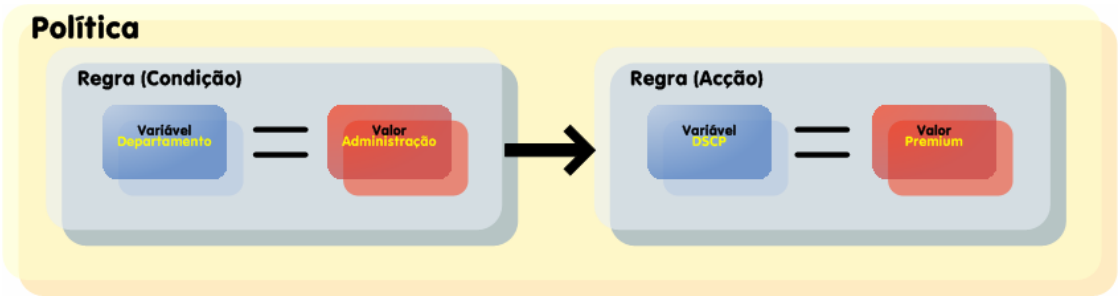


Figura 3.7 – Estrutura genérica de uma política

Cada variável faz referência a um qualquer objecto pertencente ao directório, ou a objectos específicos incluídos nos respectivos *schemas* comparando-os com os valores

indicados, e dando lugar à acção caso a condição anterior seja verdadeira. Toda esta informação terá de ser guardada no repositório de uma forma lógica que permita o seu uso e reutilização de uma forma coerente. Surgem assim os repositórios de políticas dentro do repositório geral, de onde por referenciação se podem usar as políticas guardada ou alguns dos seus subconjuntos. Como veremos mais adiante esta reutilização, dependendo da forma como cada implementação usa o repositório, pode não só guardar apenas políticas completas, mas também regras, condições, acções, associações, grupos, *roles*, subconjuntos, etc.

Outra importante característica do modelo de informação normalizado pelo IEFT e aqui aplicado é o conceito de abstracção e herança que permite também a reutilização de determinadas classes de informação por herança de níveis de abstracção superiores. Vejamos de que formas tais políticas ficam guardadas no repositório.

A representação de políticas em LDAP, segue os modelos de informação definidos primeiro pelo RFC 3060 [2], *Policy Core Information model* (PCIM) e posteriormente complementado pelas extensões e melhorias introduzidas pelo RFC 3460 [3] - *Policy Core Information Model Extensions* (PCIME). Estes Modelos de Informação são a base do RFC 3703 [6] e RFC 4104 [7], os quais definem uma implementação do modelo de informação para o directório LDAP. Assim, convém salientar no âmbito destes documentos, nomeadamente nos dois últimos, o que de mais importante há a considerar na representação das políticas em repositórios LDAP.

O método de representação baseia-se num conjunto de classes que implementam o modelo de informação PCIM. A correspondência entre as entidades do Modelo de Informação e as classes necessárias em LDAP é apresentada na Tabela 3.2.

Modelo de Informação (PCIME)	Classes LDAP
PolicySet	pcelsPolicySet
PolicyGroup	pcelsGroup pcelsGroupAuxClass pcelsGroupInstance
PolicyRule	pcelsRule pcelsRuleAuxClass pcelsRuleInstance
SimplePolicyCondition	pcelsSimpleConditionAuxClass
CompoundPolicyCondition	pcelsCompoundConditionAuxClass
CompoundFilterCondition	pcelsCompoundFilterConditionAuxClass
SimplePolicyAction	pcelsSimpleActionAuxClass
CompoundPolicyAction	pcelsCompoundActionAuxClass
PolicyVariable	pcelsVariable

	pcelsVendorVariableAuxClass
PolicyExplicitVariable	pcelsExplicitVariableAuxClass
PolicyImplicitVariable	pcelsImplicitVariableAuxClass
PolicySourceIPv4Variable	pcelsSourceIPv4VariableAuxClass
PolicySourceIPv6Variable	pcelsSourceIPv6VariableAuxClass
PolicyDestinationIPv4Variable	pcelsDestinationIPv4VariableAuxClass
PolicyDestinationIPv6Variable	pcelsDestinationIPv6VariableAuxClass
PolicySourcePortVariable	pcelsSourcePortVariableAuxClass
PolicyDestinationPortVariable	pcelsDestinationPortVariableAuxClass
PolicyIPProtocolVariable	pcelsIPProtocolVariableAuxClass
PolicyIPVersionVariable	pcelsIPVersionVariableAuxClass
PolicyIPToSVariable	pcelsIPToSVariableAuxClass
PolicyDSCPVariable	pcelsDSCPVariableAuxClass
PolicyFlowIDVariable	pcelsFlowIDVariableAuxClass
PolicySourceMACVariable	pcelsSourceMACVariableAuxClass
PolicyDestinationMACVariable	pcelsDestinationMACVariableAuxClass
PolicyVLANVariable	pcelsVLANVariableAuxClass
PolicyCoSVariable	pcelsCoSVariableAuxClass
PolicyEthertypeVariable	pcelsEthertypeVariableAuxClass
PolicySourceSAPVariable	pcelsSourceSAPVariableAuxClass
PolicyDestinationSAPVariable	pcelsDestinationSAPVariableAuxClass
PolicySNAPOUIVariable	pcelsSNAPOUIVariableAuxClass
PolicySNAPTypeVariable	pcelsSNAPTypeVariableAuxClass
PolicyFlowDirectionVariable	pcelsFlowDirectionVariableAuxClass
PolicyValue	pcelsValueAuxClass pcelsVendorValueAuxClass
PolicyIPv4AddrValue	pcelsIPv4AddrValueAuxClass
PolicyIPv6AddrValue	pcelsIPv6AddrValueAuxClass
PolicyMACAddrValue	pcelsMACAddrValueAuxClass
PolicyStringValue	pcelsStringValueAuxClass
PolicyBitStringValue	pcelsBitStringValueAuxClass
PolicyIntegerValue	pcelsIntegerValueAuxClass
PolicyBooleanValue	pcelsBooleanValueAuxClass
PolicyRoleCollection	pcelsRoleCollection
ReusablePolicyContainer	pcelsReusableContainer pcelsReusableContainerAuxClass

	pcelsReusableContainerInstance
FilterEntryBase	pcelsFilterEntryBase
IPHeadersFilter	pcelsIPHeadersFilter
8021Filter	pcels8021Filter
FilterList	pcelsFilterListAuxClass

Tabela 3.2 – Correspondência do Modelo PCIME e as Classes LDAP. [7]

Entre as classes da Tabela 3.2, destacam-se por exemplo as entidades pcelsRule, pcelsSimpleConditionAuxClass, e pcelsSimpleActionAuxClass que implementam as regras e os seus casos particulares das condições e acções para regras simples. Para condições e acções que não as elementares, existem as classes correspondentes pcelsCompoundConditionAuxClass e pcelsCompoundActionAuxClass que por associação permitem o uso de várias regras sobre uma condição ou acção.

As associações entre classes são implementadas de quatro formas possíveis [7]:

- **Por referenciação com DN:** Através de atributos dentro de uma classe que fazem referência a um objecto através do seu respectivo DN.
- **Por DIT Containement (*Directory Information Tree*):** Ou seja por aninhamento da classe subordinada sobre outra classe do nível superior da hierarquia no directório.
- **Por agregação de Classes Auxiliares:** Agregando uma classe auxiliar à classe estrutural principal, acrescentando dessa forma mais o conjunto de informação da classe auxiliar.
- **Pelo uso de Classes de Associação e DN's:** Usando classes específicas para associação e atributos que fazem também referência a DN.

A Tabela 3.3 indica quais as classes e os atributos usados para as relações de associação mais importantes do modelo PCIME [7]:

Associação PCIME	Classes e atributos da associação em LDAP
PolicySetComponent	pcelsPolicySetComponentList em pcelsPolicySet e pcelsPolicySetDN em pcelsPolicySetAsociation
PolicySetInSystem	DIT Containment e pcelsPolicySetDN em pcelsPolicySetAsociation
PolicyGroupInSystem	DIT Containment e pcelsPolicySetDN em pcelsPolicySetAsociation
PolicyRuleInSystem	DIT Containment e pcelsPolicySetDN em pcelsPolicySe-

	tAsociation
PolicyConditionStructure	pcimConditionDN em pcelsConditionAssociation
PolicyConditionInPolicyRule	pcelsConditionList em pcelsRule e pcimConditionDN em pcelsConditionAssociation
PolicyConditionInPolicyCondition	pcelsConditionList em pcelsCompoundConditionAux-Class e pcimConditionDN em pcelsConditionAssociation
PolicyActionStructure	pcimActionDN em pcelsActionAssociation
PolicyActionInPolicyRule	pcelsActionList em pcelsRule e pcimActionDN em pcel-sActionAssociation
PolicyActionInPolicyAction	pcelsActionList em pcelsCompoundActionAuxClass e pcimActionDN em pcelsActionAssociation
PolicyVariableInSimplePolicyCondition	pcelsVariableDN em pcelsSimpleConditionAuxClass
PolicyValueInSimplePolicyCondition	pcelsValueDN em pcelsSimpleConditionAuxClass
PolicyVariableInSimplePolicyAction	pcelsVariableDN em pcelsSimpleActionAuxClass
PolicyValueInSimplePolicyAction	pcelsValueDN em pcelsSimpleActionAuxClass
ReusablePolicy	DIT containment
ExpectedPolicyValuesForVariable	pcelsExpectedValueList em pcelsVariable
ContainedDomain	DIT containment ou pcelsReusableContainerList em pcelsReusableContainer
EntriesInFilterList	pcelsFilterEntryList em pcelsFilterListAuxClass
ElementInPolicyRoleCollection	DIT containment ou pcelsElementList em pcelsRoleCollection
PolicyRoleCollectionInSystem	DIT Containment

Tabela 3.3 – Associações do modelo PCIME e as classes LDAP. [7]

A modelação das políticas e a forma com elas são associadas e representadas hierarquicamente nos repositórios é relativamente flexível [3] permitindo por exemplo aninhar regras sobre regras, condições sobre condições, etc. Por esse motivo, a estruturação das políticas sobre o directório pode seguir diferentes caminhos dependendo das regras definidas ao nível da aplicação. Como exemplo, repare-se que o uso de classes auxiliares e o aninhamento de classes sobre classes de uma forma hierárquica está relacionado com questões de desempenho no uso do directório [7].

Sem querer descrever o próprio RFC 4104 neste trabalho, valerá a e pena indicar apenas um breve exemplo dos vários que o documento descreve de como o processo de agregação poderá ocorrer incluindo a reutilização e o uso de condições compostas. [7]

A Figura 3.8 ilustra o exemplo. Trata-se de uma regra, onde as suas condições e acções específicas são usadas do Repositório X.

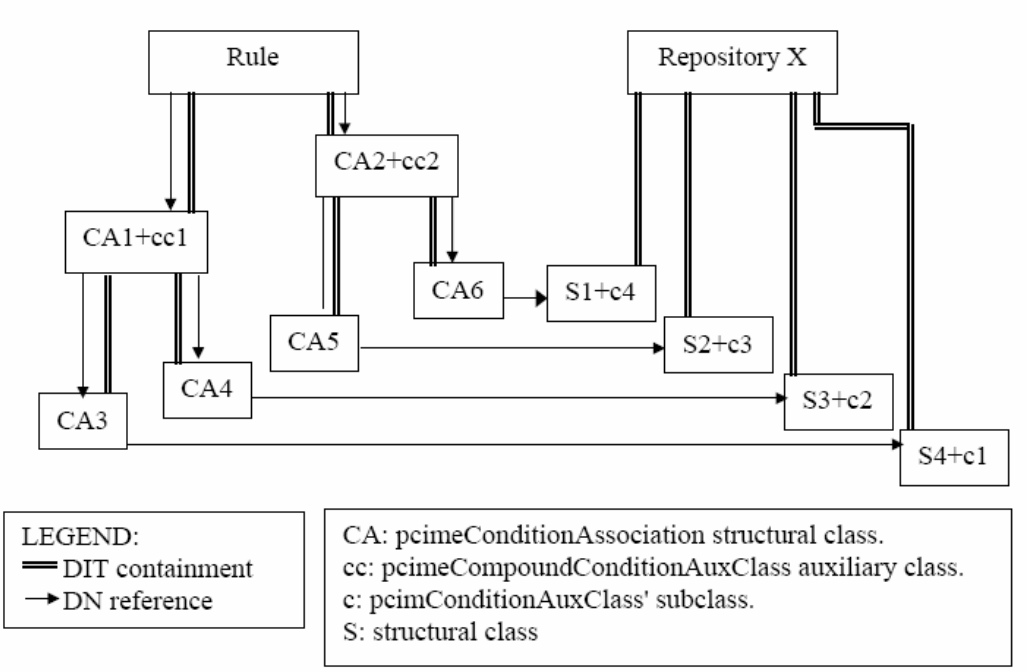


Figura 3.8 – Condições e acções compostas na construção de regras [79].

As classes estruturais pcimeConditionAssociation (CA), aqui aninhadas sobre as regras, e a seguir aninhadas ou referenciadas sobre elas próprias para as condições específicas, instanciam as condições compostas. O uso de aninhamento ou referenciação por DN depende se o uso é directo sem reutilização, ou por reutilização respectivamente.

A reutilização das regras, condições, acções, inclusivamente variáveis e valores permite agilizar a administração das políticas facilitando a sua organização no repositório.

O exemplo serviu para ilustrar a construção da regra com condições compostas reutilizadas do repositório. Os RFCs indicados detalham o processo completo com todas as suas particularidades.

3.2.6 O Directório CORDENA

No âmbito deste trabalho foi seleccionada a distribuição OpenLDAP para a implementação LDAP. Assim, sobre esta distribuição e após os serviços estarem instalados nomeadamente a base de dados embebida Berkley DB, foi necessário configurar a instância do servidor onde, entre outras configurações, se define quais os *schemas* que implementa. Foram incluídos em primeiro lugar os seguintes *schemas* elementares:

- ***core.schema***
    - Classes e atributos básicos definidos nos RFC 2252 a 2256
    - RFC 1274 (uid/dc)
    - RFC 2079 (URI)

- RFC 2247 (dc; dcObject)
- RFC 2587 (PKI)
- RFC 2589 (Dynamic Directory Services)
- RFC 2377 (uidObject - Information *schema* itens)

- ***cosine.schema* :** Com base no RFC 1274 - Cosine Internet X.500 *schema* .
- ***inetorgperson.schema*** : Definido no RFC 2798.
- ***nis.schema* :** Com objectos dos RFC 2252 e RFC 2307.

De seguida foram criados de raíz os seguintes *schemas* de acordo com os RFC publicados.

- ***cordenacim25.schema*** - De acordo com o DMTF C.I.M. Version 2. Serviu para implementar os objectos do topo da hierarquia do modelo PCIME.
- ***cordenapcim.schema*** - De acordo com o RFC 3703.
- ***cordenapcime.schema*** - De acordo com o RFC 4104.

A sua representação está patente em anexo, onde são apresentados os *schemas* usados sob a plataforma OpenLDAP.

Com a instância de base de dados preparada e configurada com os *schemas* adequados, o directório foi povoado com alguma informação oriunda o esquema organizacional descrito anteriormente e representado em árvore na Figura 3.9.

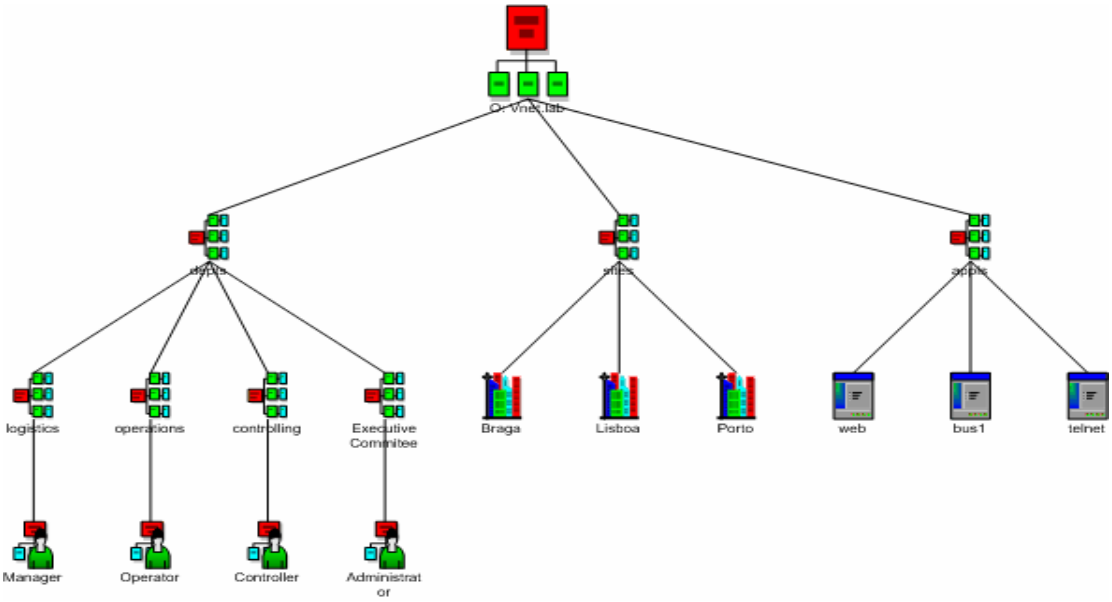


Figura 3.9 – Árvore organizacional do directório CORDENA.

Com esta estrutura completa foram testadas algumas estruturas de objectos e representadas algumas políticas de teste.

3.2.7 Notas Finais

O objectivo desta secção foi dar a entender qual a base organizacional e técnica usada, assim como uma visão geral sobre os mecanismos principais usados para guardar políticas e outros objectos.

Estamos agora com uma visão mais clara sobre a informação que circula por dentro deste modelo. A representação das políticas no directório e a forma como são guardadas é um ponto crítico para qualquer implementação que desta abordagem tire partido.

Os detalhes adicionais do modelo de informação de cada objecto enunciado aqui, serão introduzidos em secções seguintes por alguns deles estarem muito ligados às formas de tradução apresentadas. De seguida vamos analisar as operações da plataforma CORDENA e o fluxo de processos alimentado pelos modelos de informação que acabamos de analisar.

3.3 Fluxo de Processos e Operações

De forma a materializar as vantagens até agora evidenciadas por esta plataforma, vejamos que operações ou processos devem ser garantidos ou aconselhados numa perspectiva de utilização final.

A Figura 3.10 apresenta o fluxo de processos lógicos da plataforma para a versão COPS-PR, destacando os processos pela camada em que ocorrem, incluindo o interface, a camada de rede (PDP e PEP), e o PMT. Uma cópia aumentada é apresentada em anexo para facilitar a leitura.

Trata-se de uma abordagem genérica e que pretende focar alguns pontos importantes, nomeadamente os processos iniciais de instalação e configuração, como por exemplo o caso das funcionalidades de auto-descoberta da infra-estrutura, assim como de outras necessidades quotidianas relacionadas com as funções de gestão e visualização das políticas aplicadas. Por outro lado numa perspectiva mais arquitectural, abre alguns dos processos de rede que ocorrem entre o PDP e PEP baseado numa perspectiva de *Provisioning*, e algumas funções internas que a plataforma deve compreender. De seguida serão salientadas as considerações mais importantes cada fase incluída no diagrama.

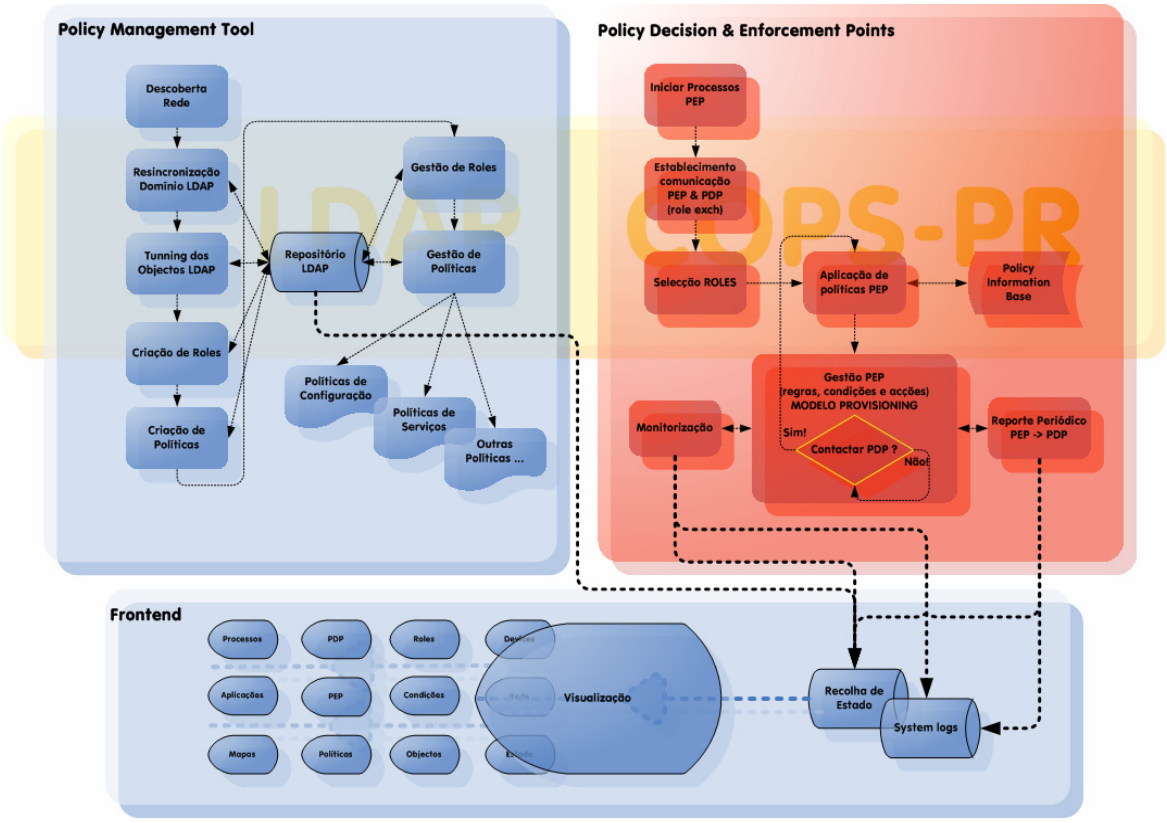


Figura 3.10 – Fluxo de processos e operações

É importante salientar antes de mais que o modelo não contempla uma linguagem própria para especificação de políticas. Essa área de investigação está a evoluir, e está claramente fora do âmbito do presente trabalho.

É no entanto importante entender que os blocos constituintes que têm sido detalhados ao longo deste trabalho, e que o serão ainda mais do decurso do presente Capítulo, são o produto final guardado sob o directório de objectos da organização e de todas as políticas e das suas partes constituintes. Todo este conjunto é passível de ser gerido através de uma linguagem própria que facilite a interacção com o repositório, ou de uma forma radicalmente diferente através de ferramentas gráficas, o que seria naturalmente necessário analisar, nomeadamente nas suas vantagens e desvantagens, e das capacidades integráveis nas fases de desenvolvimento dos interfaces apropriados.

De seguida serão salientadas as considerações mais importantes de cada fase incluída no diagrama da Figura 3.10, assim como algumas considerações gerais e não menos importantes como por exemplo a auditoria ou a integração.

3.3.1 Configurações iniciais

Por mais automático que seja o ambiente, haverá sempre que proceder à preparação da plataforma e à sua instalação inicial requerendo-se nesta fase que seja o mais transparente possível fazendo um equilíbrio entre a complexidade inevitável, o conhecimento

necessário para a execução das tarefas, e a simplicidade tentando sempre manter a operação o mais expedita possível.

3.3.2 Descoberta da rede

Uma das funcionalidades interessantes que existem implementadas em muitas ferramentas de gestão de redes é a descoberta automática dos nós da rede. Trata-se de uma característica catalizadora da aplicação, por permitir muito rapidamente começar a testar a ferramenta, mesmo que os dados necessitem ainda de alguma limpeza e ajuste.

Numa possível implementação, essa função de catalizador é muito importante, especialmente neste caso. Desta forma coleccionam-se todos os objectos da infra-estrutura numa primeira fase e posteriormente procede-se à limpeza e ajuste de propriedades, eventualmente com o auxílio de macros já pré-ajustadas.

3.3.3 Ajuste e verificação dos objectos LDAP

Uma vez que muitos dos objectos podem não conter todas as informações necessárias para implementar o modelo CORDENA, é necessário estender os *schemas* LDAP aos objectos principais. Esta fase reflecte essa operação, assim como a "limpeza" de informação sem valor acrescentado, repetida, etc. Atribuição de *roles* por agrupamento é também uma tarefa que pode ser entendida como fazendo parte deste conjunto.

Funcionalidades como selecção simultânea de vários objectos, e processos automatizados de gestão devem ser usados. Muitas das melhores funcionalidades de gestão do directório LDAP, encontram-se em aplicações próprias para o efeito de onde se pode facilmente extrapolar as funcionalidades de gestão deste tipo de conteúdos. [134]

3.3.4 Criação e gestão de *roles*

A criação de *roles* parte muito da própria visão do administrador de rede.

Para evitar ter a sensibilidade como mais uma variável de utilização do ambiente, são sugeridas algumas *roles* iniciais para garantir que pelo menos um conjunto inicial de funcionalidades é sugerido e algumas usadas. O conjunto sugerido é o seguinte:

- *roles* de Infra-estrutura
    - WAN
    - LAN

  - DMZ
  - Centro de Dados
  - Gateway Internet
  - *Router*s
  - Servidores
  - Postos de trabalho

- *roles* Organizacionais
  - Todos os utilizadores
  - Utilizadores do Dept. Financeiro
  - Utilizadores do Dept. Produção
  - Utilizadores do Dept. Logística
  - Administração

- *roles* por criticidade de Processo de Negócio
  - Processos críticos
  - Processos prioritários
  - Processos normais
  - Processos de baixa prioridade

As *roles* por processos de negócio podem ser bastante importantes na definição das particularidades de cada processo de negócio. Para lá das necessidades específicas, muitas outras utilizações desta funcionalidade podem ser equacionadas. Vejamos um exemplo distante: Nas acções decorrentes de um Plano de Continuidade de Negócio, ou de Recuperação de Desastre, algo que na maior parte dos casos obriga a uma fastidiosa inventariação e classificação dos processos de negócio e das respectivas infra-estruturas, o uso de *roles* pode desde logo facultar uma visão imediata e fiável de todas as partes envolvidas nos processos mais ou menos críticos, bastando para isso configurar *roles* por criticidade para o negócio.

Posteriormente as *roles* poderão ser usadas para os mais variados fins, podendo até ter aplicação temporária para determinado projecto finito no tempo, bastando para isso fazer as respectivas atribuições de objectos, dispositivos, etc às *roles* pretendidas.

Permitem acima de tudo ter uma visão de todos os objectos que fazem parte de uma determinada função, à qual podemos aplicar de uma só vez o conjunto de políticas adequado.

3.3.5 Criação e Gestão de Políticas

A criação de Políticas tem subjacente a criação de regras, condições e acções, podendo ser todas elas reutilizáveis ou não.

Por essa razão, o povoamento do repositório de políticas deverá começar de "baixo para cima" ou seja, constituindo um conjunto base de variáveis, valores, agregações e associações para regras, de regras condições e acções, e daqui para grupos, conjuntos, e repositórios internos de políticas.

As primeiras a serem aplicadas serão as Políticas de Configuração, as quais permitirão por o ambiente a trabalhar de uma forma normal, em tudo idêntica à anterior. Posteriormente, avança-se para a criação de políticas avançadas e mais complexas, partindo sempre do simples para o complexo, preferencialmente por reutilização de políticas mais abrangentes.

Um ponto muito importante e aqui subjacente diz respeito aos elementos constituintes das condições e acções, pois na sua quase totalidade dependem da fase de ajuste e verificação dos objectos LDAP descrita atrás. Quanto mais detalhada e correcta for a caracterização dos objectos do directório, maiores serão obviamente as capacidades da plataforma na gestão da infra-estrutura.

A manutenção das políticas deve ser auxiliada pela visualização, e pela verificação instantânea do que está aplicado à rede e do que não está, assim como o efeito real das políticas aplicadas.

A existência de uma linguagem de gestão e programação de políticas pode em grande medida extrapolar as vantagens deste modelo por acrescentar as capacidades de *scripting*, e de criação de automatismos capazes de encadear condições e acções, e evitando a visão dos blocos constituintes internos.

3.3.6 Gestão dos processos de rede

Uma vez tendo toda a plataforma CORDENA pronta a funcionar, é útil deter alguma forma de controlo sobre os processos de rede, que ocorrem nos PDPs e nos PEPs. Sugere-se atenção nas seguintes funcionalidades:

- **Arranque de processos**: As tarefas de arranque da plataforma devem ser explícitas por questões de segurança. A passagem para a gestão pelas políticas deve ser clara e facilmente constatável.

- **Sincronização de políticas**: Face a uma grande alteração de políticas, ou porventura a uma dúvida generalizada acerca do que está ou não em funcionamento, ou ainda face a algum tipo de incidência no sistema, é importante deter alguma forma de resincronização da infra-estrutura com as políticas estabelecidas. Dependendo da dimensão da rede, esta actualização pode ser controlada eventualmente pela aplicação parcial e sequencial das *roles* de infra-estrutura definidas. Não pode ter qualquer implicação em termos de indisponibilidade ou falha de qualquer ordem para os utilizadores finais.
- **Modos de Espera:** Se por qualquer motivo nomeadamente falha, determinada política apresenta um comportamento estranho, ou antes, as acções e condições que a constituem, provocaram por exemplo implicações graves numa parte substancial da rede, deve haver possibilidade de regredir ao estado imediatamente anterior, e de colocar as políticas em causa num estado latente de espera, até se apurar a causa do problema.
- **Monitorização**: Os processos de monitorização são essenciais para se verificar se o sistema está em funcionamento pleno, e se não está onde reside o problema principal. Para isso é necessário monitorizar uma série de itens, processos, dispositivos, equipamentos, etc, assim como que políticas estão a ser aplicadas, se com sucesso ou não, etc.
- **Reporte Periódico**: Com uma periodicidade que faça sentido, levando em linha de conta os ciclos de análise dos administradores de rede e essencialmente de acordo com os *Service Level Agreements* (SLA) e *Service Level Specifications* (SLS), definidos com os respectivos clientes. Exemplos de reporte adicionais poderão incluir, informações estruturadas ou *ad-hoc* retiradas do repositório LDAP (podendo ser via ferramenta externa); Relatórios das Políticas em uso e em que objectos e *roles*; Relatório de Políticas inactivas; Relatórios de Actividade por PDP, PEP; Relatórios de Erros, e muito mais.

Muitas outras funcionalidades poderão fluir do uso da plataforma, no entanto estas são já uma ideia interessante das ferramentas que deverão existir para controlar o ambiente da melhor forma.

3.3.7 Visualização

A visualização abarca quase todos os pontos falados ao longo deste Capítulo.

Embora não seja crítico o seu apuro, uma vez que não está no âmbito deste trabalho apurar as especificações de um ambiente de trabalho ou um *frontend* para a plataforma, devem ser indicadas algumas considerações.

Há que distinguir que a visualização e percepção dos efeitos da aplicação das políticas, podem ver-se através de outras ferramentas externas especificamente desenhadas e

talhadas para monitorização de infra-estruturas. Desde ferramentas *Open Source* como um MRTG, NAGIOS, até outras ferramentas comerciais de renome. Todas elas permitem através de SNMP, ou protocolos proprietários, colher uma imagem em linha do estado da rede, em termos de larguras de banda usadas, serviços em funcionamento ou parados, verificação de limites de uso predefinidos, e muito mais.

Haverá no entanto que acrescentar técnicas de visualização próprias para Gestão de políticas e verificação de estado no que à Plataforma de Gestão de Políticas diz respeito.

Vejamos alguns exemplos de ferramentas de visualização que se recomenda que sejam integradas:

- Mapa da rede com indicações visuais da atribuição de políticas.
- Ferramenta de *frontend* gráfica baseada em janelas, e também em texto.
- Listagem das politicas em cada PEP
- Mapa de apresentação visual e on-line dos PEPs e dos respectivos PDP com que interagem.

O próprio directório pode ser uma ajuda crucial para esta representação pois toda a informação que detém, e as relações entre elas permitem criar mapas lógicos e até com algum carisma geográfico que podem surpreender nos quatro pontos indicados.

A visualização é também uma função que pode ganhar bastante com a modelação da topologia de gestão com base em patamares de informação com crescentes níveis de detalhe. Vendo a infra-estrutura do domínio de gestão a diferentes níveis de detalhe podemos gerir facilmente descendo ou subindo e aplicando ao nível respectivo as políticas adequadas ao contexto visualizado, fazendo uma pequena comparação com o que uma ferramenta de navegação rodoviária com auxilio de GPS permite ao ver um mapa em diferentes níveis de aproximação e com as operações que mais sentido fazem em cada nível. (ex. ao nível do percurso faz sentido planear a rota, ao nível da rua é útil procurar uma alternativa local)

Um mecanismo desta natureza é a solução para uma visualização eficaz e funcional.

Apenas com o intuito de esboçar o que poderá ser uma aproximação à interface principal da plataforma, apresenta-se o seguinte protótipo gráfico com os vários componentes do *frontend*. A Figura 3.11 apresenta as notas acerca das principais componentes e a sua utilização. Uma cópia aumentada é apresentada em anexo para facilitar a leitura.

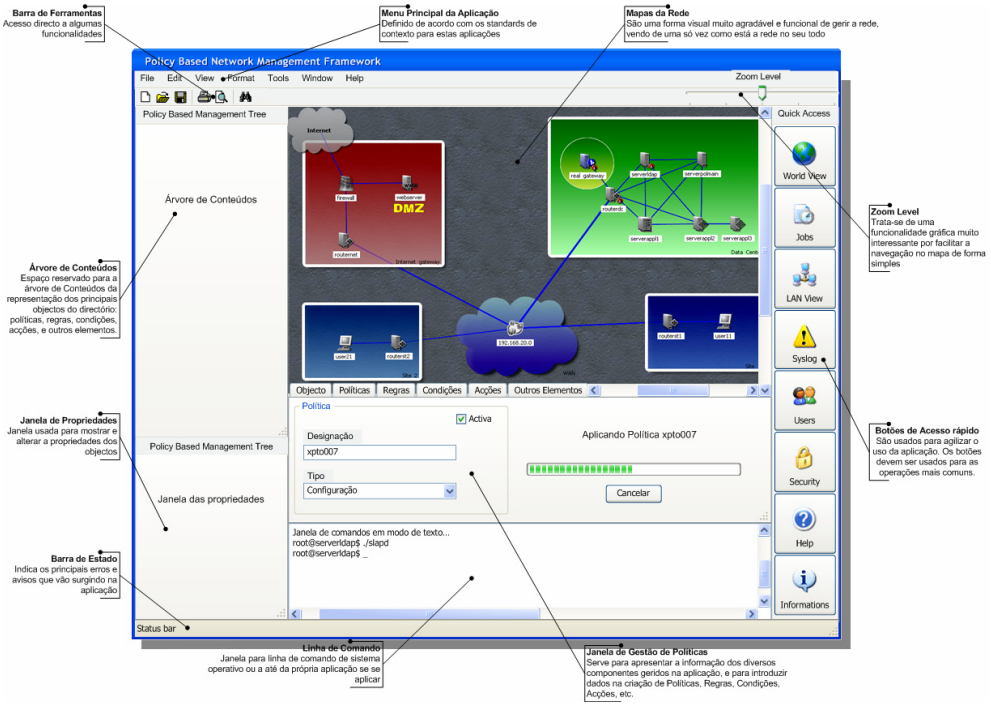


Figura 3.11 – Protótipo gráfico da interface com o utilizador

Não se especificam cada uma das componentes em detalhe desde o menu, árvore de conteúdos, etc, deixando tais definições ao critério da fase de desenvolvimento da aplicação, uma vez que depende bastante desta fase técnica.

O único ponto que se realça é a importância de seguir as regras e definições para o desenvolvimento de interfaces gráficas de acordo com os ambientes gráficos em que a aplicação irá correr, nomeadamente, Gnome, KDE, MS Windows, GNUStep, ou outros. Este ponto é crucial por fazer com que o utilizador já habituado a utilizar determinado gestor de janelas não necessite de formação específica e não seja surpreendido com funções e funcionalidades alocadas em menus estranhos, ou então não consiga encontrar as funções que procura.

3.3.8 Monitorização e auditoria

Adicionalmente aos requisitos de visualização indicados no ponto anterior, onde se mencionam já algumas formas de monitorização, acrescentam-se agora as necessidades de geração de logs para registo de actividade, e recomenda-se o uso de alarmista para activação de processos de suporte em caso de falha. Os *logs* são a impressão digital da aplicação.

Neste caso e sem sacrificar muito os processos de escrita, deve ser garantida a recolha da informação necessária e suficiente para constatar que o sistema está a executar o que é suposto, para permitir o rasteio de erros, problemas e funcionalidade da aplica-

ção, quer durante a fase desenvolvimento, quer posteriormente em produção. O formato poderá ser a norma de texto ASCII.

Uma possível especificação para o *logging* da plataforma poderia ser a indicada de seguida:

Contexto

- Data/hora
- Identificação das *roles* em causa
- Identificação das políticas em causa
- Identificação das instâncias repositório LDAP , PMT, PEP, PDP
- Identificação da operação
- Descrição do evento

Código

Poderá ser acrescentado um código de mensagem com sete dígitos para referenciar a actividade do sistema, cuja constituição começa com três dígitos que identificam o tipo de mensagem conforme se indica de seguida, e acaba com três algarismos sequenciais que indicam o erro, aviso ou informação a que a mensagem se refere.

Identificação do tipo de mensagem

- INF - Mensagens de informação
- AVI - Mensagens de aviso
- ERR - Mensagens de erro

No caso das mensagens de erro, os números indicados podem ser usados para distinguir a criticidade dos problemas que ocorrem da seguinte forma:

Codificação da Criticidade

- ERR-100 até ERR-199: Erros de baixa Criticidade!
- ERR-200 até ERR-299: Erros de Média Criticidade!
- ERR-300 até ERR-399: Erros de Alta Criticidade!
- ERR-400: Bloqueio completo da infra-estrutura (Panic!)

Para lá do *log*, os processos de monitorização normal podem ainda ser controlados por ferramentas externas específicas para a análise e confirmação das acções executadas pela plataforma, via SNMP por exemplo. Podem ser acrescentados à monitorização os processos que de alguma forma têm haver com a plataforma, por exemplo, os processos *slapd* do servidor LDAP, os *deamons* da plataforma no servidor PMT, etc....

3.3.9 Integração

Deve ser facilitada a integração com outros sistemas externos, nomeadamente no que diz respeito ao repositório de objectos do domínio de gestão. Usando a norma LDAP como mínimo denominador comum, a integração com outros sistemas que embora sendo proprietários de terceiros, garantem a compatibilidade com LDAP, deixa em aberto a possibilidade de ampliar os benefícios desta plataforma a sistemas de rede baseados por exemplo em Microsoft Active Directory, Lotus Domino – Notes, ou Novell eDirectory, para citar apenas alguns.

3.3.10 Segurança

Embora não é um ponto fulcral na análise deste trabalho de investigação, o que não acontece numa implementação em produção, é uma temática demasiado importante para não merecer um espaço no trabalho numa perspectiva de requisitos.

As garantias de autenticidade na autenticação e da não violação dos conteúdos na comunicação e transporte de políticas, são características importantes que têm que existir em qualquer implementação do género, por exemplo com recurso a funcionalidades adicionais de encriptação, basilares neste tipo de comunicação com informações importantes de negócio se for o caso.

Muitas nas trocas de informação com o directório podem usufruir das várias possibilidades oferecidas de raíz por estes, desde autenticação KERBEROS, encriptação da informação com vários algoritmos disponíveis, etc.

3.3.11 Notas finais

A secção finda apresentou um resumo das principais operações que a plataforma deve suportar. O seu valor acrescentado repercute-se na necessidade de entender a arquitectura como sendo uma solução que responda também a estes requisitos aqui identificados. Com o Modelo de Informação subjacente e os processos e operações a suportar, concluímos um conjunto de especificações de requisitos aos quais a plataforma dará resposta com a arquitectura detalhada na próxima secção.

3.4 Arquitectura CORDENA

Desde o Repositório de Políticas, até aos elementos de rede, passando por todas as componentes da arquitectura, vamos agora detalhar a arquitectura por detrás da plataforma CORDENA, assim como os princípios de base.

As primeiras quatro secções descrevem os processos e as ideias nucleares do modelo, e o detalhe do processo de tradução de entidades de alto nível em variáveis de rede.

As secções seguintes apresentam para cada entidade as suas particularidades e a forma como foram moldadas no modelo de informação, ou seja quais as classes seleccionadas para a sua representação. Embora todos os *schemas* estejam descritos em anexo, são apresentados em cada caso os quadros resumo de cada entidade oportunamente quando se detalha o seu uso.

Por fim far-se-á uma incursão sobre os princípios gerais acerca da interface entre os PDPs e os PEPs e na forma como se entende que estão todas as condições criadas para transferir parte da "inteligência da rede" para os elementos do último patamar, aliviando os processos de gestão nos PDPs e aumentando o desempenho de uma forma geral. Para fechar o ciclo, esta secção apresentará no final a visão geral da arquitectura e dos principais blocos constituintes.

3.4.1 Servidor de Políticas

Depois de descrito o conjunto de processos e operações que a plataforma suporta, vejamos agora com que meios internos o servidor de políticas implementa as funções nucleares do ambiente descrito e de que forma estão relacionados para atingir os objectivos propostos.

Como já vimos há duas grandes entidades no modelo: o Repositório de Políticas que guarda grande parte da informação do sistema, e o Servidor de Políticas que descrevemos de seguida e que manipula todo o sistema desde a interacção com o Repositório, até ao envio, recepção e tratamento dos pedidos de rede.

A Figura 3.12 ilustra as partes lógicas constituintes do servidor, onde sobre um quadro de fundo que representa o esqueleto aplicacional da plataforma, se unem todos os módulos subjacentes numa disposição modular e adaptável. O quadro integra módulos e adaptadores que interagem com diversas entidades externas ao sistema colhendo o estado da rede, as políticas a aplicar, enviado informação para rede, etc.

Comecemos pelo módulo LDAP responsável pela interacção com o Serviço de Directório que age como o Repositório de Políticas na rede. O módulo é responsável pelas

normais operações de criação, eliminação, alteração, enfim de gestão dos objectos do repositório, sejam eles políticas, regras e variáveis associadas a políticas, ou qualquer outro objecto do repositório. O seu alcance é definido mais por questões de segurança e perfis de acesso, do que de falta de qualquer funcionalidade, pois a implementação da API respectiva é relativamente simples. A implementação desta funcionalidade passa para escolha e uso do API LDAP correspondente ao Directório em uso. (ex. OpenLDAP) [48].

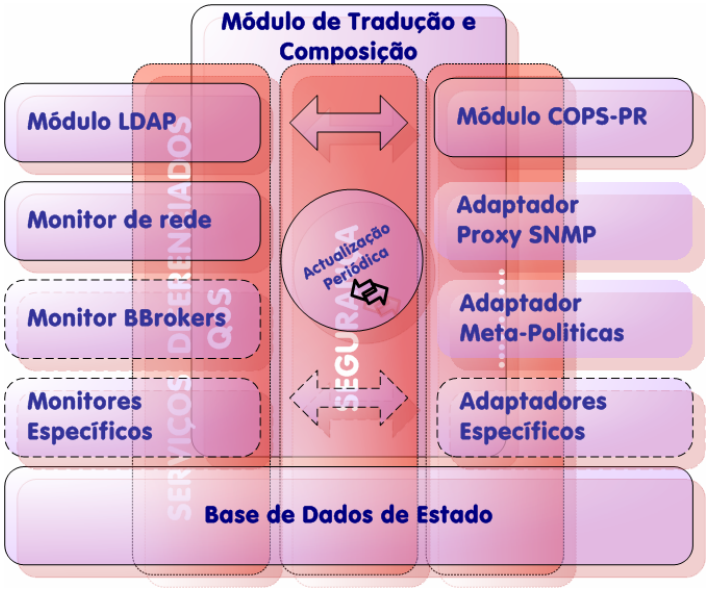


Figura 3.12 – Arquitectura do Servidor de Políticas

Os módulos monitores de rede encarregam-se de manter a informação de estado da infra-estrutura actualizada localmente de forma a manter um base de informação de acesso rápido em *cache*. A sua utilização é ilimitada. Como conjunto mínimo de monitorização deverá ser garantida a existência de um monitor NTP para garantir a existência de uma fonte fidedigna de data/hora. Para lá deste, é de igual forma importante ter um monitor que tem a informação necessária e suficiente dos principais pontos da rede para apurar por exemplo do estado de congestão da rede, de forma a poder desencadear a aplicação das políticas adequadas.

A estrutura modular destes monitores é importante de forma a poder integrar novas fontes de informação, inclusivamente algumas não previstas inicialmente, como por exemplo um monitor para recolha de informações de um *bandwidth broker* tendo dessa forma informação acerca da admissão num determinado domínio. Com o conjunto de monitores de rede indicados e o módulo adaptador LDAP, temos assim acesso a toda a informação necessária para a criação e gestão das políticas, falta agora transferir tais políticas para rede. Para isso contamos com o módulo COPS-PR ou outros adaptadores específicos como por exemplo sinalização NSIS. Estes módulos servem para depositar numa estrutura de dados definida pela tecnologia usada, o resultado da composição efectuada após transformar os objectos em variáveis de rede. Estas informações são o

ponto de partida para a interacção com os PEPs no envio e recepção das mensagens de distribuição e controlo de políticas.

Por exemplo, o módulo COPS-PR é responsável por criar as mensagens COPS-PR REQ, DEC, etc de acordo com os RFC 2748 e 3084 e com as informações já recolhidas dos processos de tradução que ocorreram no Repositório e no Servidor de Políticas. Da mesma forma é responsável por responder às solicitações da rede com as mensagens adequadas, quando um equipamento pede uma decisão ao servidor.

Ainda no âmbito da interacção com a rede, refira-se a inclusão no diagrama de um adaptador para a compatibilização do envio de políticas para equipamentos sem suporte para tal, através de um *proxy* de políticas, e ainda um adaptador para meta-políticas por necessitar de operações específicas de *parsing* ligeiramente diferentes pelas alterações introduzidas nos PIBs.

Segue-se um dos módulos mais importantes do servidor. O módulo de tradução e composição dedica-se a colar todas as partes indicadas anteriormente no intuito de face às necessidades identificadas nas políticas a aplicar à rede, iniciar a construção as mensagens. A Figura 3.12 divide este bloco em três partes que representam as diversas aplicações da plataforma de Gestão baseada em Políticas. Aqui indicam-se as aplicações de QoS e Segurança. Estes blocos são transversais a todos os outros. Poder-se-á dizer de uma forma simples que une as informações do directório LDAP com as informações dos monitores adequados para resolver as condições patentes nas regras das políticas que terá de aplicar.

Finalmente uma breve alusão ao círculo central que representa o processo sempre activo à escuta das alterações da rede para despoletar a aplicação de políticas por alteração de estado da infra-estrutura.

O estado de todo o sistema do servidor de políticas reside numa instância local de uma base de dados representada no último patamar do diagrama. Esta base de dados contém entre outros objectos, uma *cache* de informação oriunda do repositório principal com o dicionário de variáveis e valores mais usados.

3.4.2 Mecanismos de tradução

Desde o repositório LDAP até aos nós da rede onde as políticas são aplicadas há principalmente dois passos onde os mecanismos de tradução podem actuar.

Em primeiro lugar do Repositório para o Servidor de Políticas, passo (2) da Figura 3.13, e do servidor para os nós da rede, passo (3).

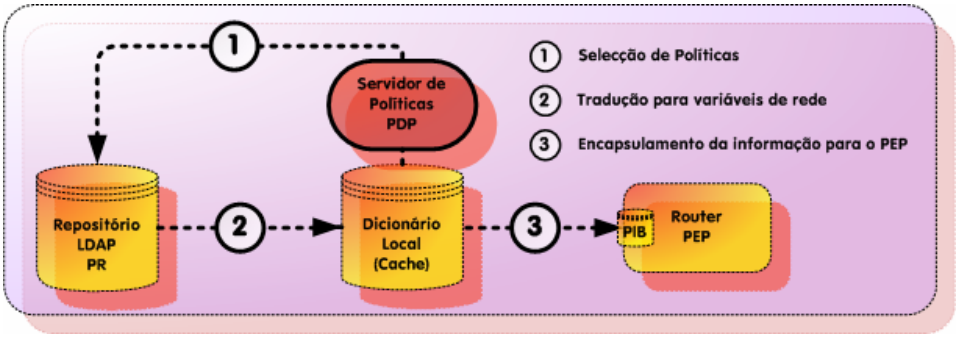


Figura 3.13 – Principais passos de tradução de objectos

No primeiro passo são traduzidas variáveis do repositório em variáveis de rede, usadas na avaliação das condições e criação das acções no servidor de políticas. No segundo passo com todas as variáveis já traduzidas e guardadas localmente, falta apenas enviar para a rede as mensagens compostas de acordo com a pilha protocolar usada e o PIB definido.

Dependendo do protocolo usado na comunicação das políticas com os PEPs, diferentes métodos podem ser usados para o encapsulamento e construção das mensagens e para a gestão do PIB, normalmente implementados com as respectivas APIs.

De acordo com o modelo de informação descrito e usando agora os objectos principais guardados no repositório, vejamos o que queremos traduzir.

A Tabela 3.4 indica qual o resultado da tradução que se pretende, ou antes, que variáveis de rede poderão representar o objecto indicado. Na prática pretende-se chegar aos resultados da coluna mais à direita com o mínimo esforço possível de processamento, de interacções na rede, de ocupação de largura de banda e de fiabilidade no sentido em que deve ser garantido que as variáveis da coluna da direita reflectem pela certa as variáveis indicadas à esquerda.

É claro que tanto as variáveis de alto nível como as variáveis de rede estão à partida limitadas pelo âmbito deste trabalho. As primeiras pelo conjunto escolhido e definido no modelo de informação, e as segundas por serem apenas as necessárias para gerir a marcação de pacotes via DSCP, o necessário para enquadrar o modelo sob a aplicação de QoS via Serviços Diferenciados sem entrar no detalhe das várias entidades de controlo do modelo DiffServ, tal como indicado oportunamente acerca do âmbito do trabalho.

De qualquer forma é importante salientar que dependendo das aplicações de PBNM em causa, outras variáveis podem entrar em jogo com diferentes mecanismos de tradução. A aplicação da plataforma CORDENA à gestão de segurança por exemplo, poderia usar directamente a variável "Utilizador" e "Departamento" sem necessitar de tradução num determinado contexto, porque são variáveis necessárias directamente na configuração das *firewalls*.

A solução deve ser tal que permita da forma mais transparente, a aplicação destas funcionalidades a redes já constituídas e em produção, criando o mínimo de turbulência possível na implementação.

Variável de alto nível	**Principais problemas a equacionar**	**Variável de Rede**
Utilizador ex. gteixeira	Normalmente os endereços de rede dos utilizadores são dinâmicos mudando de acordo com a validade dos leases dos servidores DHCP. Não são portanto de carácter estático.	Endereço IP ex. 10.128.120.120
Departamentos Ex. Departamento de Produção	Os utilizadores de um Departamento podem estar em redes completamente distintas e sofrem também do carácter temporário e dinâmico dos endereços IP.	Aplicação de políticas por *roles* ou por colecções de IP's.
Localizações físicas ex.Edifício A	Uma localização física pode ser constituída por várias subredes originando um conjunto de IP's diferentes mas oriundos da mesma localização física. As gamas podem ser mudadas e portanto é necessário haver um qualquer procedimento para evitar a desactualização.	Colecções de gamas arbitrárias de endereços IP. ex. 10.128.120.0 mask 255.255.255.0 10.140.122.24 mask 255.255.255.252 ...
Ambientes aplicacionais ex. ERP, Módulo Financeiro, Módulo Vendas, etc...	Protocolos com uma comunicação complexa podem ser difíceis de representar.	Endereço do servidor, porta do serviço e protocolo usado. ex. 10.10.10.1 : 3200 tcp
Processos de negócio ex. Processos de venda, compra, de produção, de expedição, processos de análise financeira, etc..	Se os processos estão suportados sob a mesma aplicação, a divisão por processos fica dependente ou da parte interna da aplicação, ou da análise e interpretação das mensagens de rede dos níveis 5 a 7 do modelo OSI.	Se os processos estão suportados por aplicações diferentes, aplica-se a informação dos ambientes aplicacionais, caso contrário terá de se enveredar por tratamento caso a caso. Ex. Tipo de mensagem, identificação do módulo aplicacional, ou identificação das funções de determinados objectos no directório, nomeadamente utilizadores
Referenciação temporal ex. Data/Hora, Duração	Há a necessidade de sincronizar o tempo em todo o domínio de aplicação contemplando também os fusos horários.	Neste caso não há um processo de tradução associado, mas sim a necessidade de ter a data/hora sincronizada. ex. Data/Hora, Duração
Variáveis dos Equipamentos Terminais	Localização distribuída	MIB Local

Tabela 3.4 – Resultados pretendidos com a tradução de objectos.

As próximas secções irão descrever como é composto o funcionamento de cada passo indicado, e quais as considerações mais importantes acerca de cada processo.

3.4.3 Primeira Fase de Tradução (Repositório)

Após a selecção das políticas a aplicar, inicia-se o primeiro processo de tradução. A decomposição das políticas seleccionadas em regras. Cada regra como vimos contém variáveis e valores. O primeiro passo é a tradução dessas variáveis que inicialmente nada significam para a rede, em endereços IP fixos, máscaras de rede, etc.

Para o conjunto de objectos organizacionais em análise propõe-se um modelo de tradução baseado no próprio directório, isto é, os processos de tradução para os objectos em causa ocorrem do directório e não no Servidor de Políticas, embora este receba as variáveis traduzidas e processa a todo o restante trabalho de construção e encapsulamento das mensagens. Este modelo é proposto por algumas razões importantes que se apresentam de seguida.

Se os processos de tradução ocorrerem no repositório, a troco de pouco desenvolvimento e de pequenas adaptações, teremos sempre disponível e actualizada a informação técnica para a criação e gestão de políticas. Os processos de actualização de políticas são aliviados da necessidade de verificação constante dos valores dos objectos, distribuindo o processamento e diminuindo o consequente aumento do tempo de resposta, pois de outra forma os valores iriam ser verificados no momento. Desta forma, toda a informação traduzida está já disponível tratando-se apenas de correr um processo de consulta simples sobre os objectos pretendidos.

A mapeamento dos valores traduzidos e dos objectos de negócio correspondentes é bastante simples, pois cada entidade no directório terá as suas variáveis de rede necessárias para o encapsulamento das mensagens de rede em COPS-PR ou outro qualquer protocolo de transporte de políticas. Assim quando o Servidor de Políticas necessita da informação traduzida, ela está pronta e sempre disponível.

Para que isto seja possível propõe-se uma estrutura dos objectos em LDAP com um *schema* mais rico. De uma forma geral cada objecto do directório deverá conter três partes principais: Cabeçalho, Informação Estática e Informação Dinâmica. Vejamos a Figura 3.14. O primeiro bloco de informação é apenas o que cada objecto já tem, tratando-se dos tradicionais campos Nome e Descrição entre outros.

Estes campos serão as "chaves de procura" do servidor de políticas, ou por outras palavras as variáveis de alto nível visíveis na construção das políticas. Aqui os campos designados no *schema* como "*MUST*", ou seja, campos obrigatórios devem ser usados como índices de procura para evitar as pesquisas sobre campos "*MAY*" não obrigatórios.

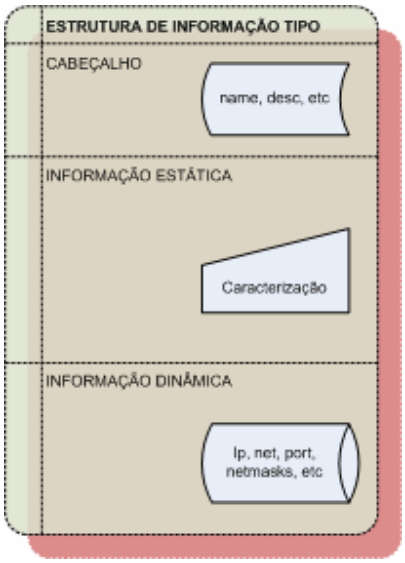


Figura 3.14 – Estrutura proposta dos objectos em LDAP

Este ponto é particularmente importante na identificação das chaves a indexar na configuração do servidor. O segundo bloco de informação contém as restantes propriedades que completam a caracterização normal do objecto. O último bloco de informação é acrescentado neste modelo e guarda as variáveis de rede necessárias para a construção das mensagens de rede.

Aplicar este modelo a qualquer servidor LDAP é relativamente simples, pois trata-se de simplesmente estender o *schema* inicial com as classes (auxiliares) necessárias.

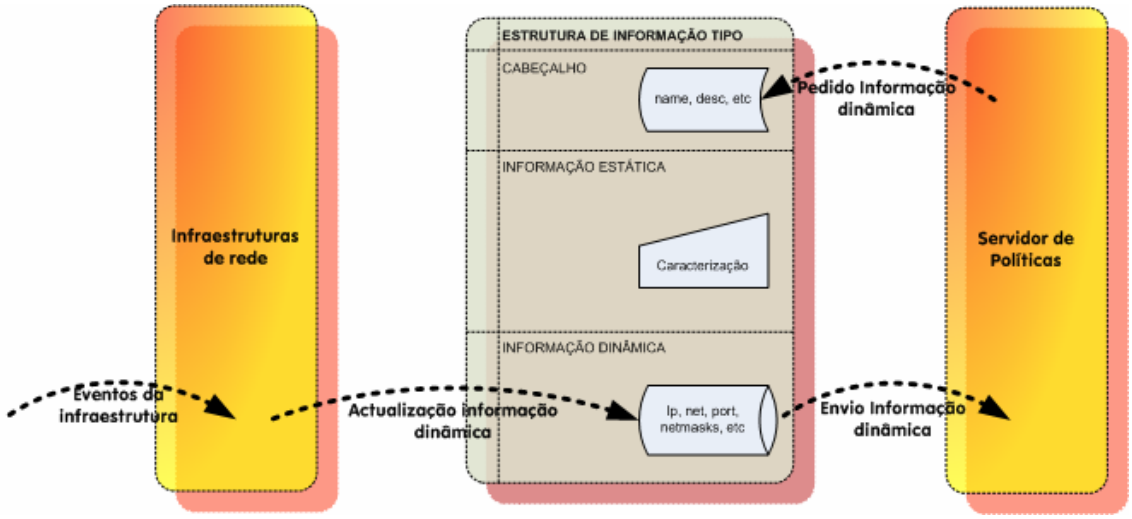


Figura 3.15 – Processos de conversão e actualização das variáveis

Por outro lado, o facto de fazer com que tais processos ocorram no directório, evita que haja *pooling* desnecessário para testar os valores traduzidos das políticas. A excepção são as alterações no directório que podem ser seguidas de uma mensagem de aviso ao Servidor de Políticas quando há uma actualização no directório para actualizar a *cache*.

Esta abordagem pretende simplificar o processo garantindo uma solução fiável e facilmente implementável numa plataforma de gestão de redes baseada em políticas.

Será importante salientar o facto de que a objectivo principal é colher da rede os valores das variáveis correctos, e não só representar estaticamente um endereço IP para um determinado utilizador. Não é definitivamente esse o objectivo. Pretende-se sim "deixar" a rede operar normalmente sem nenhum constrangimento de endereçamento fixo ou predefinido, mas sim buscar uma forma de manter as informações actualizadas no directório gerando para isso o valor mínimo de mensagens de *pooling*. Desta forma atingiremos o máximo desempenho no que respeita à actualização das variáveis na plataforma CORDENA. A Figura 3.15 esquematiza os processos de conversão aqui desenvolvidos.

Uma última nota para integrar neste modelo a abordagem reflectida no RFC 2589 [30] referente às extensões LDAP para directórios dinâmicos. No RFC 2589 são indicadas as extensões necessárias para a implementação de objectos dinâmicos para os quais se indica o objecto dinamicObject como forma de implementar esses objectos com tempo de vida limitado. Embora eventualmente esta abordagem temporal possa ser usada em alguns objectos, o carácter do modelo de informação CORDENA aposta em ter espaço no modelo de informação para guardar os últimos valores conhecidos, ao mesmo tempo que mantém sempre presente a caracterização de cada objecto através das classes que implementa e do seu perfil na organização.

3.4.4 Segunda Fase de Tradução (Servidor de Políticas)

Uma vez recebidos os valores traduzidos no Servidor de Políticas, são passados por uma zona de *caching* de forma a melhorar o desempenho de pedidos posteriores, sendo esta zona a "gare" de recepção de dados do repositório.

Vamos então ver como se processa a partir daqui a segunda fase de tradução encarregue de passar as definições das políticas para os elementos de rede. A análise apoia-se sobre a Figura 3.16 onde se tem uma percepção gráfica do exemplo de aplicação de uma política simples que altera a prioridade do tráfego para "*Premium*" caso se trate de tráfego "de" ou "para" algum dos membros do Departamento indicado.

Como se trata de um conjunto de utilizadores, o resultado recebido vai ser uma colecção de endereços IP. O outro valor recebido é a tradução de "*Premium*" para os valores do octeto correspondente do DSCP, no caso "001 111 00" ou valor 60 em decimal.

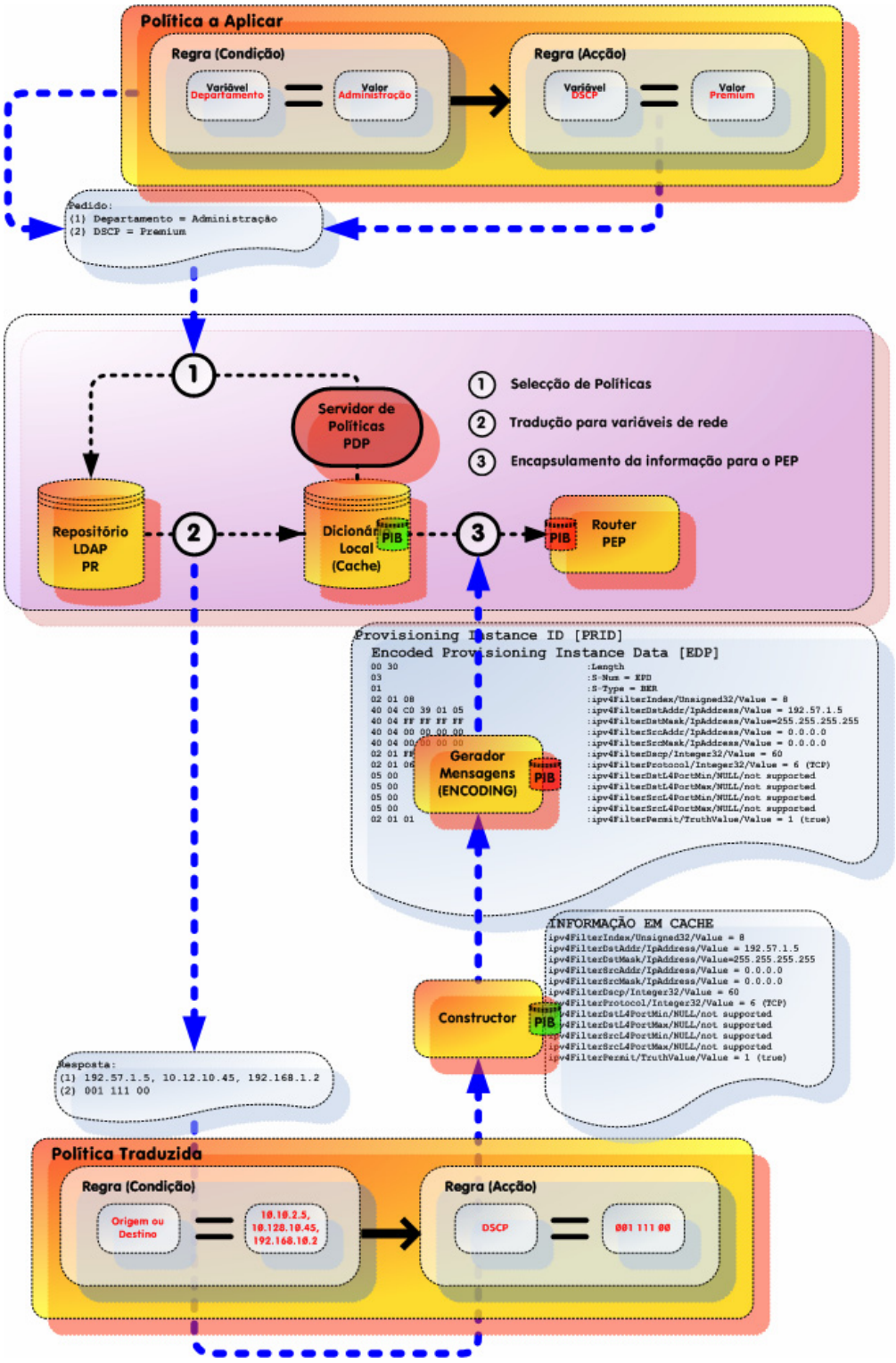


Figura 3.16 – Processos de conversão e actualização das variáveis

Com estas informações na *cache*, os processos seguintes encarregar-se-ão de construir os blocos de informação necessários de acordo com a estrutura de dados PIB correspondente. Assim, uma vez que o exemplo é baseado no uso de COPS-PR para a troca de informação com os PEPs, a partir de agora os passos são:

- **Estruturação da informação de acordo com o PIB** respectivo para Serviços Diferenciados de acordo com o RFC 3317, usando neste caso as classes (PRC) de classificadores necessários para filtrar o tráfego de acordo com as indicações da política já traduzida, e invocando o PRC respectivo para a marcação do tráfego com DSCP marker para o valor 60.
- **Geração das mensagens necessárias de acordo com o *encoding*** definido para COPS-PR. A imagem mostra um exemplo simples do resultado desse encapsulamento. Esta operação rege-se pelo RFC 3084 e representa uma mensagem com 2 objectos COPS-PR, o - *Complete Provisioning Instance Identifier* (PRID), que identifica a instância que guarda os valores a enviar ao PEP, e um objecto *Encoded Provisioning Instance Data* (EPD), isto é, que guarda os dados propriamente ditos.

O exemplo indicado apresenta um exemplo fictício retirado do RFC 3084 - ipfilterv4 apenas para demonstrar o processo.

Desta forma concluímos o processo de tradução com as seguintes fases:

- selecção das políticas a aplicar no Servidor de Políticas;
- carregamento para *cache* local dos objectos traduzidos oriundos do Repositório;
- construção das estruturas de dados de acordo com o PIB respectivo e o *encoding* das informações nos objectos de COPS-PR necessários;
- envio por exemplo de uma mensagem DEC (decisão).

Como se viu o exemplo está baseado no protocolo COPS-PR, no entanto dependendo dos módulos implementados sobre o Servidor de Políticas, outros protocolos de transporte poderão ser usados adaptando e desenvolvendo os processos de encapsulamento adequados.

3.4.5 Identidade Utilizador

Existem inegavelmente duas entidades nucleares em qualquer organização: O Utilizador, e o Departamento. Embora outros tipos de agrupamento existam em qualquer repositório, todos eles são agrupamentos de utilizadores por uma ou outra razão organizacional. Entre os dois, o Utilizador é o grão menor sendo que o Departamento será sempre um conjunto de utilizadores, por isso a análise é maioritariamente centrada no primeiro.

O perfil de informação do Utilizador no directório inclui em primeiro lugar a informação típica de registo, nomeadamente nomes, moradas, telefones, e outras informações de carácter informativo; em segundo lugar as credenciais de acesso encriptadas, perfis, autorizações e os grupos a que pertence, e por último poderá conter outras informações

dependendo das necessidades de outras plataformas aplicacionais que coabitem sob a mesma infra-estrutura. Para lá destas informações estáticas, outras dinâmicas poderão também existir que simplifiquem os processos de tradução. Nesse leque de informações são incluidos também os endereços IP, máscaras de rede, *tokens* ou outros identificadores. Estes objectos são talvez dos mais importantes e úteis do repositório, mas também os mais exigentes em termos de necessidades de actualização e de fiabilidade na solução para traduzir a identificação em localizações lógicas de rede.

Sobre eles incidirão grandes volumes de consultas dada a grande necessidade de associar configurações e acções a condições com base nos atributos: Utilizador e Departamento. Daí que a capacidade de resposta do repositório às consultas de uma forma eficaz é importante. Importante será também encontrar métodos que traduzam os nomes para endereços de rede sem acrescentar peso nas comunicações para actualizar grandes volumes de informação. Vejamos de que forma este objectivo pode ser alcançado.

Cada utilizador individual inicia a sua sessão na rede com um *login*. Este *login* é executado sobre um domínio de autenticação, por exemplo *Network Information System* (NIS), Active Directory, ou seja o repositório ou directório onde reside toda a informação da infra-estrutura.

A solução proposta para a actualização ou tradução da sua identidade para um endereço de rede, passa por depositar a informação dinâmica correspondente ao utilizador no directório nomeadamente o seu endereço IP, no momento da autenticação e após validação correcta. Assim teremos sempre disponível o último IP conhecido, até que nova alteração tome lugar, e se isso acontecer os valores devem ser actulizados.

Este processo não implica em termos de desenvolvimento grandes alterações, podendo ser facilmente integrado com os módulos *Plugabble Authentication Modules* (PAM) sobre OpenLDAP por exemplo. As classes necessárias para implementar o mecanismo ao modelo de informação correspondente ao objecto Utilizador apresentam-se na Figura 3.17.

Neste caso optou-se por indicar como classe para a informação estática a classe posixAccount por ser a indicada para implementar autenticação sobre o directório OpenLDAP num domínio Linux ou Unix pois inclui as necessárias informações da directoria pessoal, nome de utilizador na rede, etc.

Como as duas classes aqui indicadas são auxiliares, a instanciação do objecto Utilizador só se consegue tendo pelo menos uma classe estrutural, a qual indicamos ser a inetorgperson da qual é só apresentada a camada de propriedades obrigatória.

Informação Estática

General

OID 2.5.6.6

Name person

Description RFC2256: a person

Properties

Superior top

Kind Structural (0x01)

Required Attributes

- sn
- cn

Optional Attributes

- userPassword
- telephoneNumber
- seeAlso
- description

General

OID 2.16.840.1.113730.3.2.2

Name inetOrgPerson

Description RFC2798: Internet Organizational Person

Properties

Superior organizationalPerson

Kind Structural (0x01)

General

OID 1.3.6.1.1.1.2.0

Name posixAccount

Description Abstraction of an account with POSIX attributes

Properties

Superior top

Kind Auxiliary (0x02)

Required Attributes

- cn
- uid
- uidNumber
- gidNumber
- homeDirectory

Optional Attributes

- userPassword
- loginShell
- gecos
- description

Informação Dinâmica

Figura 3.17 – Classes da Entidade Utilizador

O *schema* usado fica completo com a classe person de onde é herdada alguma informação obrigatória.

Uma última nota para o atributo ipHostNumber indicado na Figura 3.18, por ser um campo multivalor de forma a permitir o registo de vários IP caso se aplique.

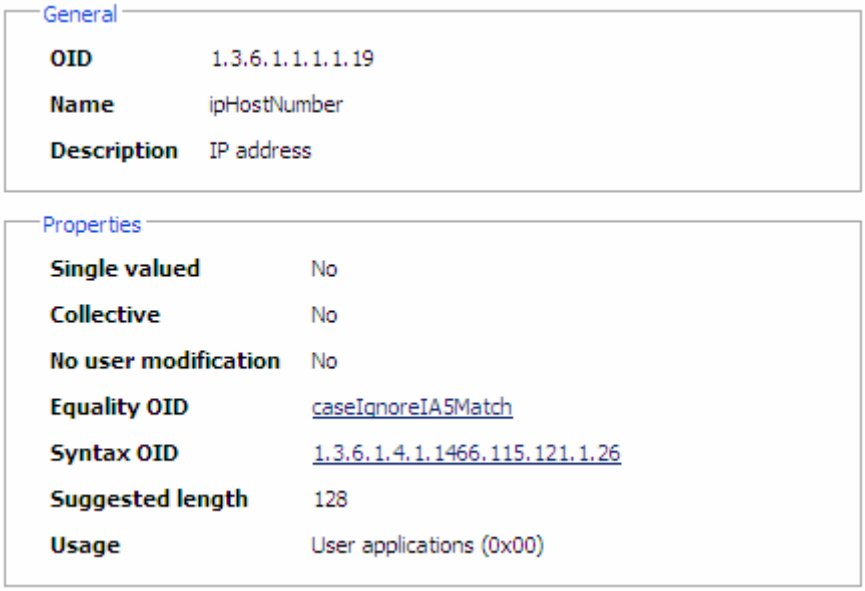


Figura 3.18 – Atributo ipHostNumber

O detalhe de todas as classes e propriedades pode ser consultada nos respectivos Anexos onde estão descritos os *schemas* utilizados.

3.4.6 Entidade Departamento

Os objectos "Departamento" em LDAP implementam uma relação de 1:N com os Utilizadores, estando estes agrupados neste modelo por Departamento. O agrupamento sob a entidade superior é designado por DIT *Containment*. Desta forma quando um Departamento é referenciado, todos os utilizadores que lhe pertencem são devolvidos.A Figura 3.19 mostra um exemplo deste tipo de agrupamento.

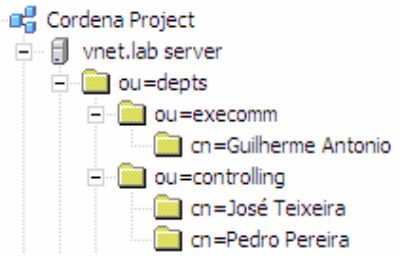


Figura 3.19 – DIT Containment - Utilizadores nos Departamentos

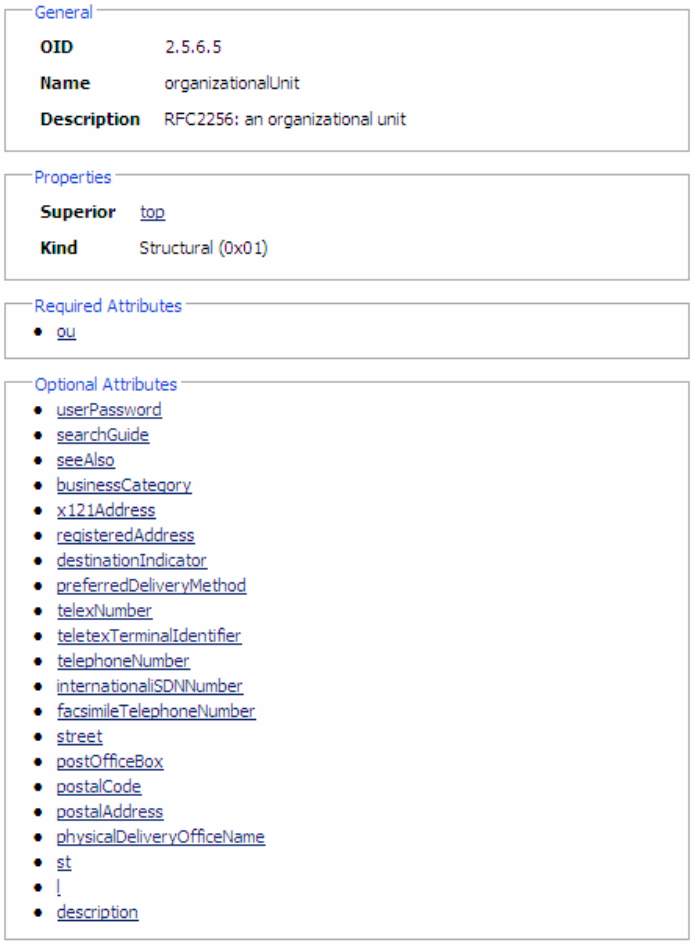


Figura 3.20 – Classe organizationalUnit

Embora o *schema* possa ser complementado com mais informação, a classe de base para este objecto é a organizationalUnit indicada na Figura 3.20.

Quando é referenciado um Departamento, todos os atributos pretendidos dos seus constituintes vão ser devolvidos, ou seja, no caso de endereços IP todos os endereços em uso por utilizadores registados.

A forma como estas informações são tratadas fica relativamente à mercê dos métodos desenvolvidos pela aplicação para usar estes recursos, com vista a alcançar o máximo desempenho e funcionalidade.

3.4.7 Localizações físicas

As localizações físicas referem por exemplo edifícios através de referências geográficas descritivas.Para este efeito, e uma vez que os dados não vão ser usados para localização georeferenciada, uma identificação descritiva é suficiente, pois de alguma forma esta "variável" terá de ser traduzida para os intervalos correctos de endereços IP, máscaras de rede, etc. Para apoiar esta tradução, outras informações relativas ao endereçamento poderão ficar aqui já guardadas, até porque contrariamente ao carácter dinâmico de um endereço IP de um utilizador, uma localização física dificilmente é dinâmica em termos de endereçamento.

Assim, acrescentar gamas de endereços IP parece à partida uma boa solução e suficientemente flexível para acolher as diversas possibilidades, uma vez que tal informação pode aceitar vários valores para registar locais que tenham várias gamas de endereçamento.

Várias estruturas de directório comerciais são apresentadas com *schemas* que contemplam na sua génese o mapeamento das topologias lógicas às físicas. O caso da *Active Directory* da Microsoft é disso um exemplo com as entidades *Sites*. De facto na instalação deste tipo de directórios, é necessário configurar a topologia de rede física e dela tirar partido para optimizar a forma como as replicações vão ocorrer. Isto indica que em situações análogas mesmo com qualquer tipo de directório, estas entidades com mais ou menos informação estão presentes. No presente caso vamos configurar a entidade Localizações que deterá as informações mais importantes guardadas.

Uma vez que contrariamente aos utilizadores, um local não muda facilmente de gamas de endereços IP, sendo esta variável um atributo que é apenas manipulado por pessoal técnico aquando de uma instalação ou remodelação geral, faz sentido mantê-la estática por referenciação no conjunto de informações da Localização.

Para organizar a informação relativa às Localizações do directório, é criado um agrupamento DIT com a classe organizationalUnit que agrupa as Localizações (*Sites* na Figura 3.23). Sobre esse agrupamento, são criadas as Localizações com o conjunto de classes: locality e pcelsIPv4AddrValueAuxClass indicadas na Figura 3.21.

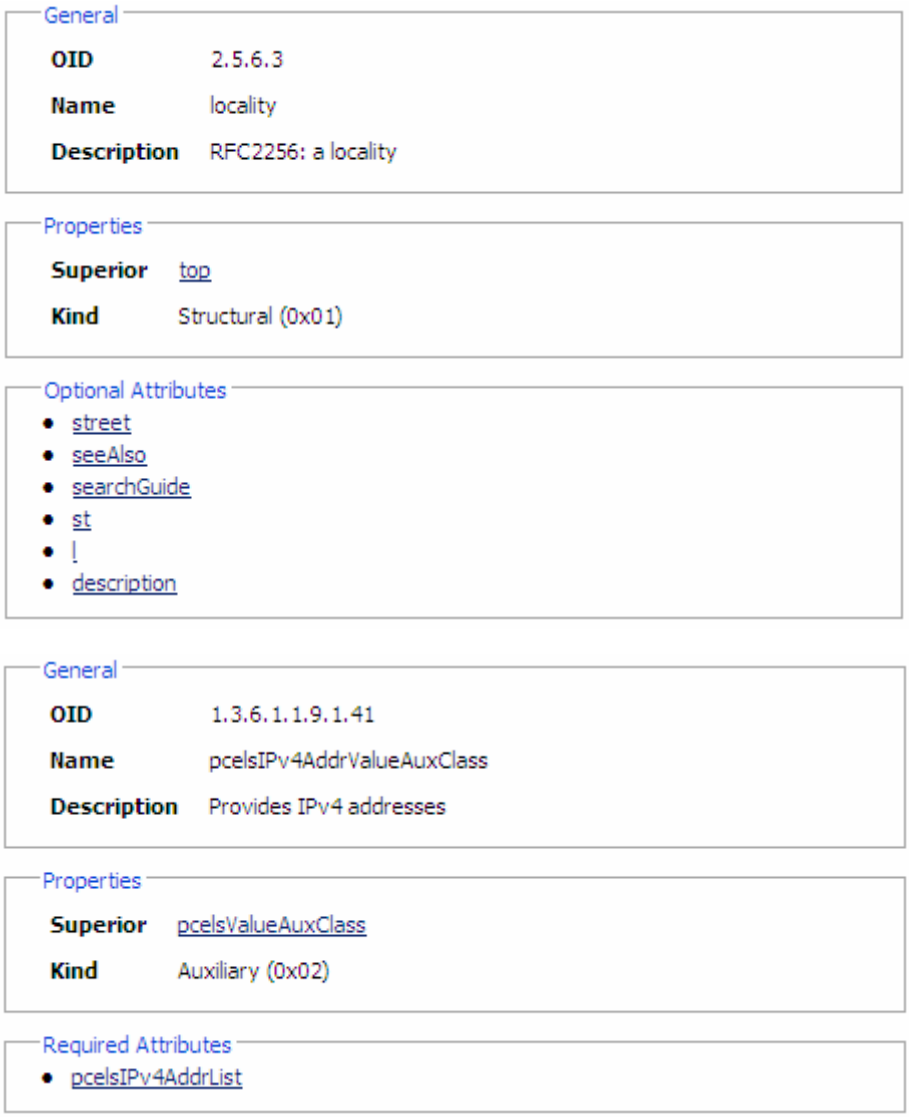


Figura 3.21 – Classes para entidade Localizações Físicas

No entanto para estender o uso das entidades Localização e também porque a classe pcelsIPv4AddrValueAuxClass não contém uma propriedade que guarda as máscaras de rede, inclui-se sob a entidade Localização, e através de DIT *Containment* as classes ipNetwork da Figura 3.22, que permitem desta forma registar uma lista de gamas de redes com os respectivos endereços IP e máscaras de rede.

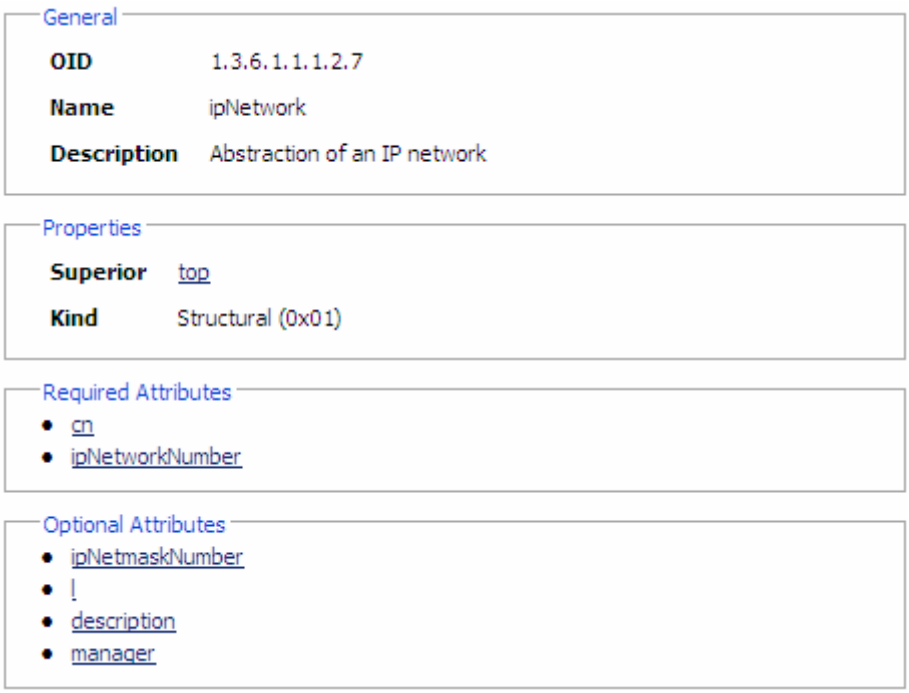


Figura 3.22 – Classe ipNetwork adicional para as Localizações

No final navegando pela estrutura de dados, o agrupamento ficará com o aspecto ilustrado na Figura 3.23, onde se indicam os "Pisos" como exemplo de duas gamas de endereços IP.

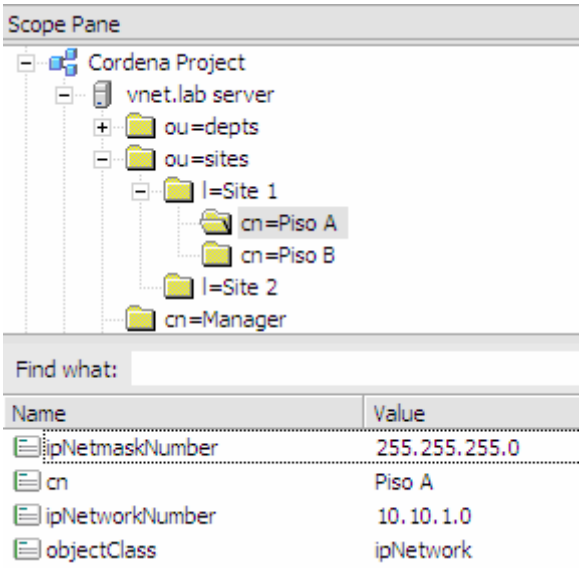


Figura 3.23 – Localizações

3.4.8 Ambientes aplicacionais

Os ambientes aplicacionais referem-se às principais aplicações de negócio e aos seus principais módulos constituintes. Não esqueçamos que o principal objectivo de determi-

nar estes objectos organizacionais e de negócio, é justamente poder usá-los como condições na definição de políticas. Fazendo apenas referências às principais aplicações em uso, teremos a capacidade para definir políticas que nos permitam por aplicação e de uma só vez gerir acessos, perfilar os recursos de rede, gerir prioridades de tráfego, etc. Poder-se-ão incluir informações como a indicação de nome, endereços de rede, portas e eventualmente indicação de prioridades caso as soluções de balanceamento de carga ou tolerância a falhas tirem partido de alguma indicação aqui guardada.

As aplicações são também entidades que não alteram as suas propriedades facilmente a não ser na sua instalação inicial e em possíveis reorganizações.

Embora já haja ambientes aplicacionais que alimentam os sistemas de directório com as suas informações nas instalações, tornando desta forma os directórios mais ricos em informação e mais integrados com toda a infra-estrutura, o facto é que ainda não é uma prática generalizada, aparecendo muitas vezes como funcionalidade opcional [141].

Na plataforma CORDENA, vamos modelar esta entidade com um conjunto de classes normais que permitem recolher as informações habitualmente mais importantes da identificação de uma aplicação. Assim a entidade "Aplicação" vai instanciar a classe applicationEntity indicada na Figura 3.24, sobre a qual apenas foi passado para opcional o atributo presentationAddress por ser já obsoleto e mais complexo no tratamento.

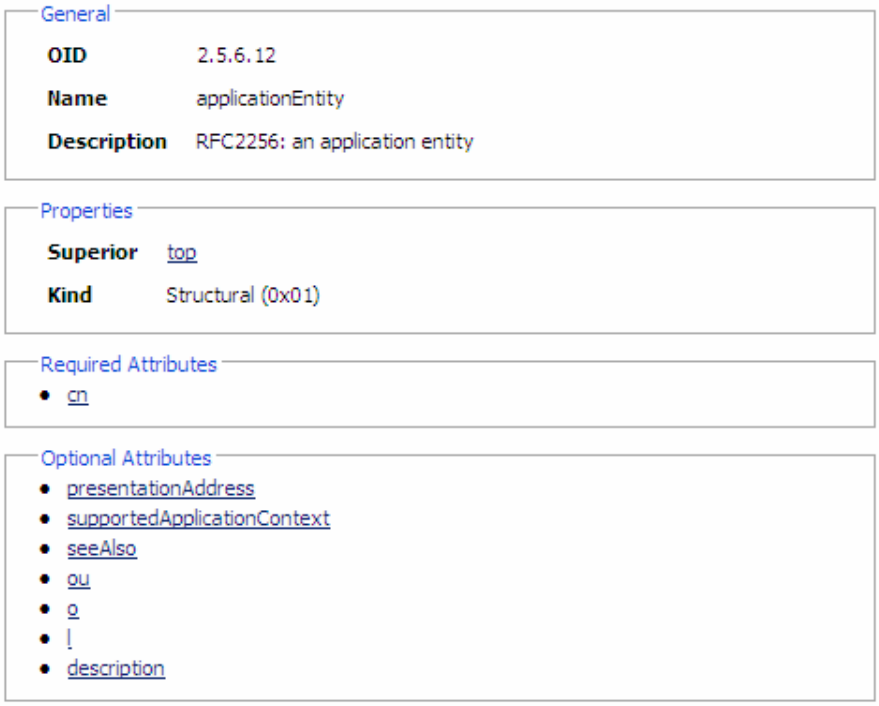


Figura 3.24 – Classe applicationEntity para Aplicações

Em sua substituição foi incluída uma subordinação de entradas das classes ipHost e ipService sob o agrupamento criado pelas instâncias de applicationEntity, classes indicadas de seguida na Figura 3.25.

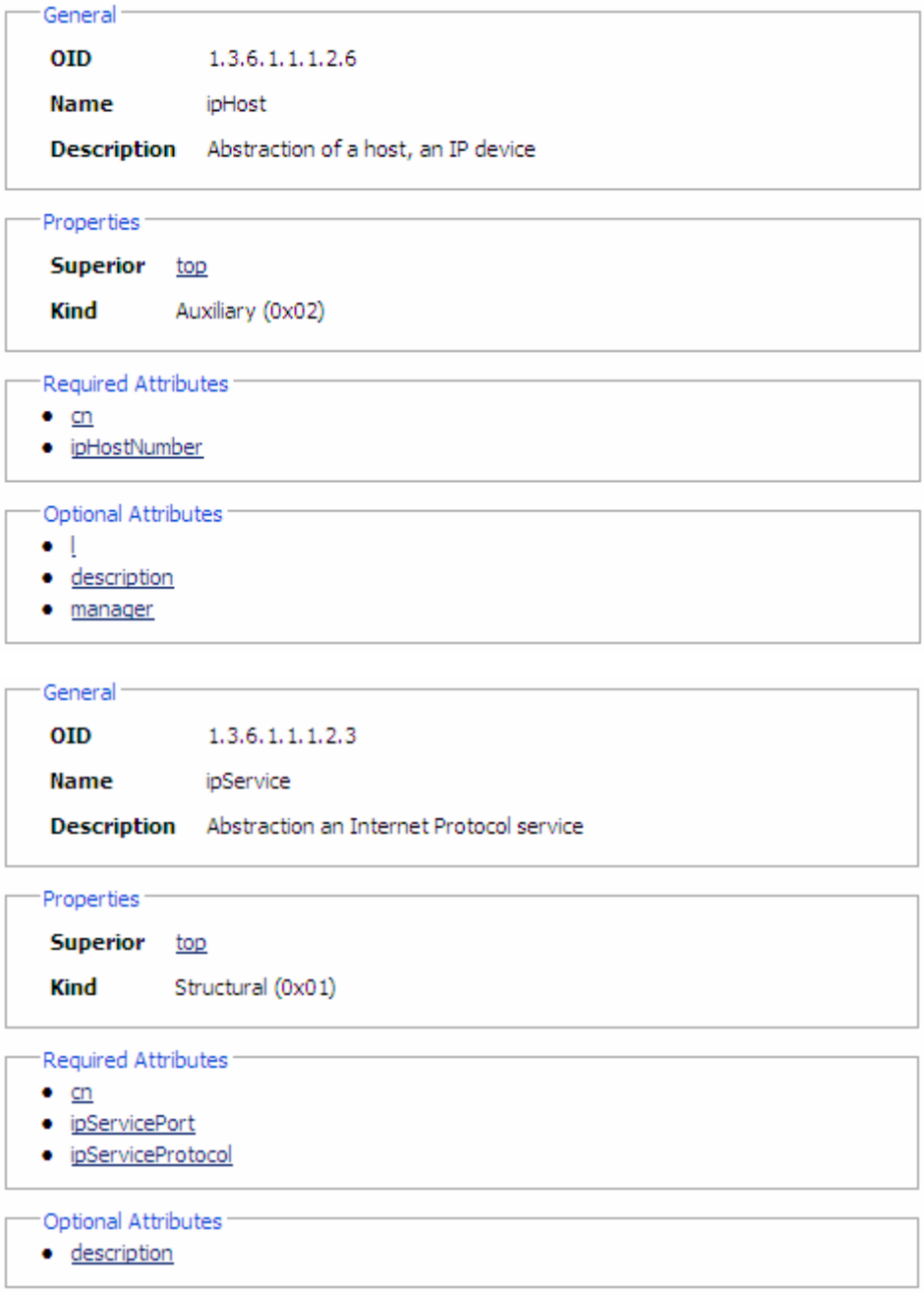


Figura 3.25 – Classes ipHost e ipService

Desta forma passa a ser possível criar várias conecções simultâneas para uma mesma aplicação, funcionalidade útil para registar *clusters* aplicacionais. Todas as aplicações estão também agregadas sob uma entidade superior baseada na classe organizationalUnit.

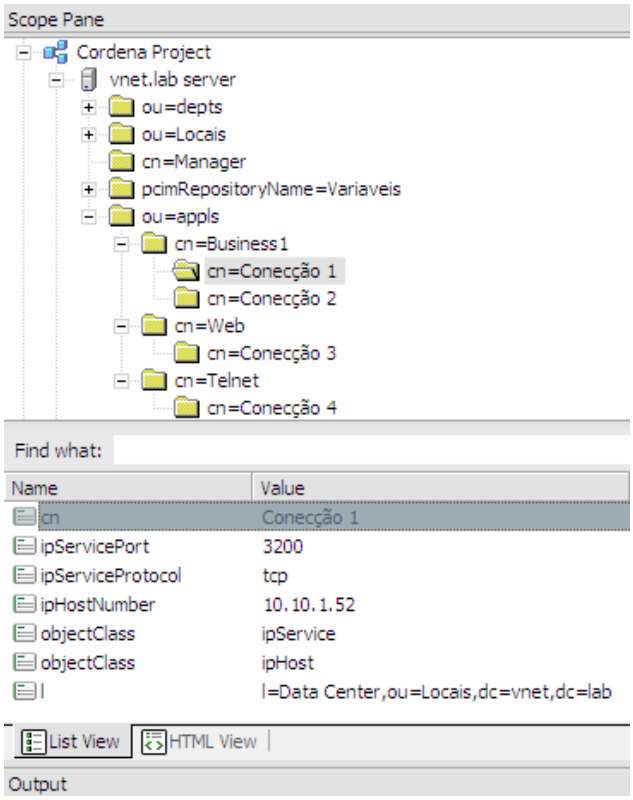


Figura 3.26 – Ambientes Aplicacionais

A Figura 3.26 mostra o ramo da árvore LDAP que guarda estas informações. De notar que o atributo "l" (localização) referencia a localização física desta aplicação. No exemplo “l= Data Center” dentro da "ou" Locais no mesmo directório.

3.4.9 Processos de Negócio

O objectivo principal ao incluir processos de negócio neste modelo é distinguir de alguma forma a parte aplicacional que suporta um determinado processo de negócio e não a aplicação como um todo, de forma a poder configurar condições por processos de negócio, ou antes, pelas partes aplicacionais que suportam um determinado processo. O objectivo é nobre mas difícil de implementar e depende em grande parte da forma como as aplicações de negócio hoje em dia são instaladas sobre as infra-estruturas.

A verdade é que cada vez mais a configuração de novas aplicações sobre uma infra-estrutura de suporte ao negócio tem de ser totalmente integrada com um serviço de directório que se assuma como um farol onde as entidades na rede consultem e depositem informações que auxiliem outras entidades a integrarem-se mutuamente. Esta abordagem passará a permitir num futuro próximo que as funcionalidades das aplicações não se esgotem dentro das suas linhas de código, mas que todas elas funcionem em matriz tirando partido das funcionalidades umas das outras.

Voltemos aos ambientes aplicacionais aqui descritos onde teremos que distinguir entre aplicações independentes usadas por um grupo bem definido de utilizadores, e aplicações modulares de suporte a vários processos de negócio de forma integrada, ou seja, os conhecidos ERPs. No primeiro caso teremos por exemplo uma aplicação específica para Gestão da Qualidade e usada apenas pelo Departamento de Controlo de Qualidade, e no segundo um típico ERP suportando vendas, compras, produção, etc.

Se no primeiro cenário a referenciação da aplicação é simples porque se materializa em informação TCP/IP fixa (IPs e portas), no segundo é mais difícil e obriga-nos a subir nas camadas do modelo OSI, ficando dependentes das capacidades aplicacionais em causa e da maior ou menor integração da aplicação com o repositório. Para a obtenção desse objectivo haverá que encontrar um "ponto de entrada" que permita a um directório e à aplicação interagirem de forma transparente num determinado contexto.

Seja com for, estaremos sempre maioritariamente perante uma de duas situações: ou o conjunto aplicacional em causa está preparado para falar com a rede algo mais que permita distinguir processos nas mensagens, ou por outro lado algo terá de ser desenvolvido à medida de forma a haver provavelmente um mecanismo de análise mensagens ao nível da sessão ou superior. Este último ponto levanta no entanto uma preocupação legítima e que deve ser tida em conta, pois a análise de padrões sobre as mensagens pode criar facilmente um grave problema de desempenho acrescentando sob a comunicação uma latência difícil de justificar. Por esta razão estas implementações devem ser criteriosamente enquadradas.

Outra possibilidade a considerar é o uso de comutadores *layer 4-7* de última geração com capacidade para gerir tráfego aplicacional, integrando-os (gestão, comutação, balanceamento de carga, etc) nos serviços de directório existentes. Desta forma o problema referido antes pode ser ultrapassado. Neste caso, estes equipamentos específicos fariam o papel de PEP com as necessárias capacidades mapeadas em variáveis locais, usadas nas políticas.

Mas voltando aos processos de negócio, e pensando de uma forma totalmente diferente, a configuração de políticas com base em condições construídas sob a identificação de determinados processos de negócio, pode ser conseguida através da utilização de "*roles*" ou "funções". Desta forma todos os objectos do directório, são "etiquetados" com os “papéis” que desempenham no todo. Embora mesmo assim, não se ultrapasse a limitação indicada nos primeiros parágrafos desta secção, ganha-se pelo menos em flexibilidade podendo identificar todas as partes constituintes da arquitectura que de alguma forma têm impacto num determinado processo. No modelo CORDENA, são apresentadas as várias possibilidades aqui levantadas.

O registo é baseado na classe nuclear applicationProcess que guarda a identificação do processo de negócio, agrupada por uma unidade organizacional para o efeito como se pode ver na Figura 3.27 e 34.

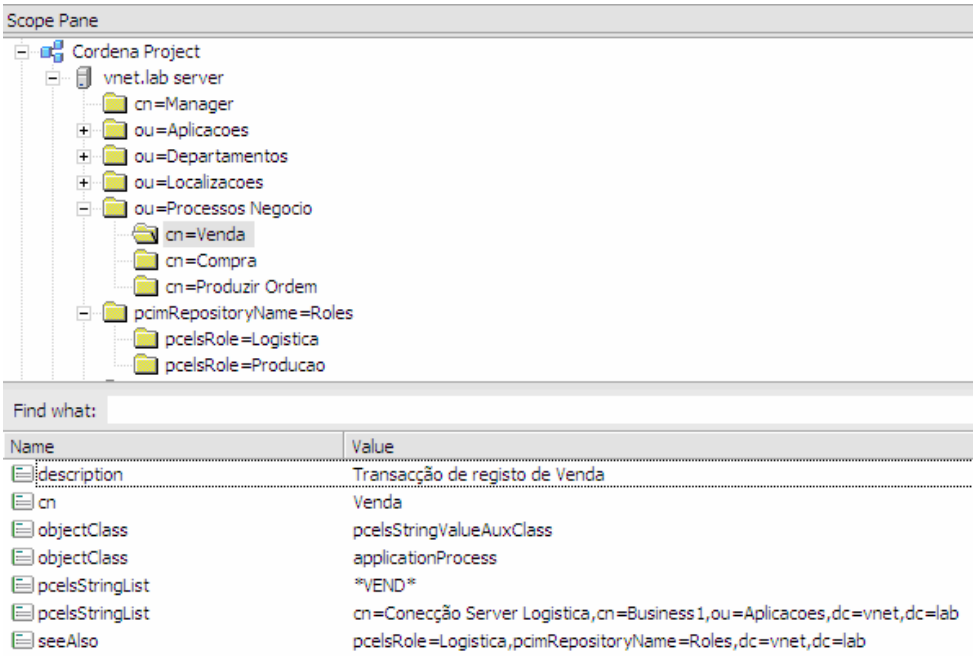


Figura 3.27 – Processos de Negócio.

Como referência às várias formas de identificação referidas em cima, é acrescentada a classe pcelsStringValueAuxClass da Figura 3.29, permitindo a instanciação de valores compostos por cadeiras de caracteres, que aqui é usado para poder guardar quer padrões para caracterizar mensagens, quer DN para referenciar uma conexão a uma aplicação de suporte.

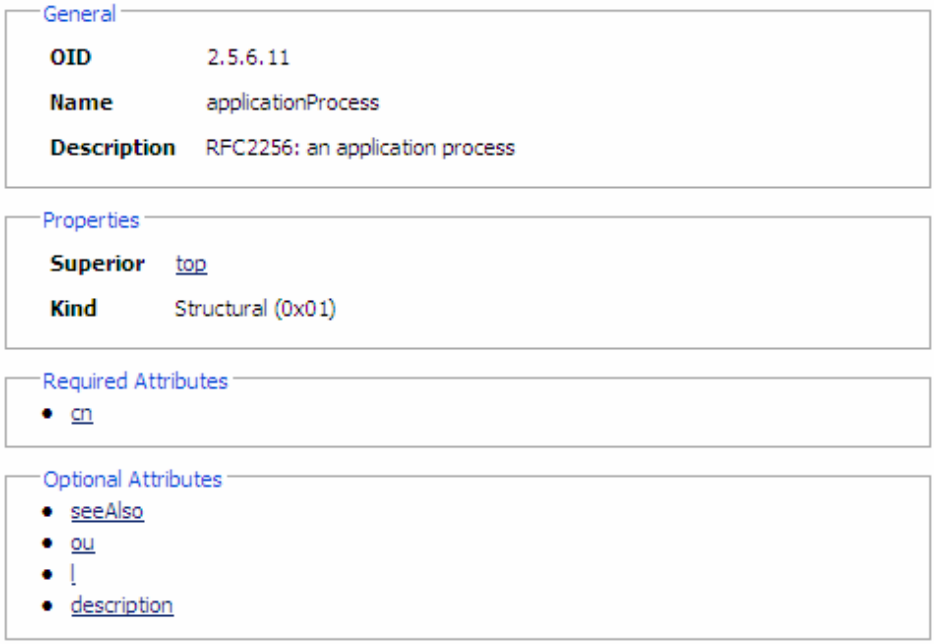


Figura 3.28 – Classe applicationProcess

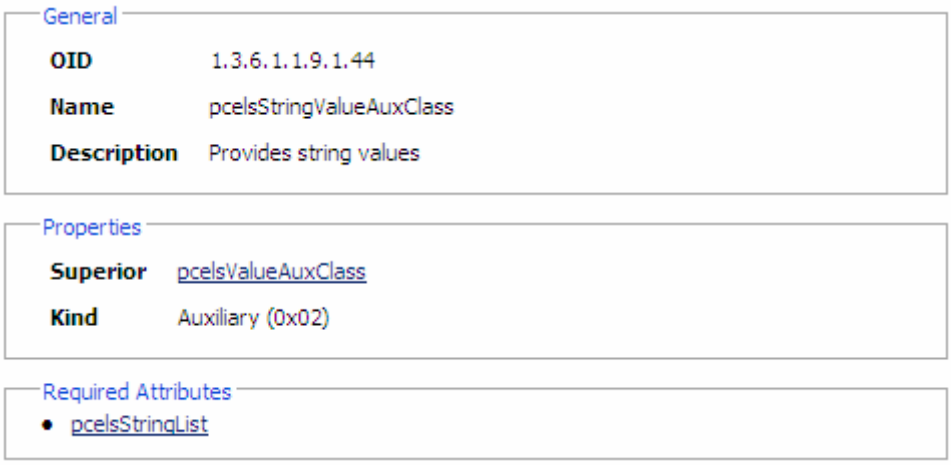


Figura 3.29 – Classe pcelsStringValueAuxClass

É claro que estes dados servem apenas para exemplificar o modelo. Numa implementação concreta tais classes poderão ser mais detalhadas e até criadas à medida.

Há ainda a salientar o atributo "*See Also*" do tipo DN da classe applicationProcess, e que aqui foi usado para fazer referência a um conjunto de *roles* ou funções dentro do mesmo contexto de negócio.

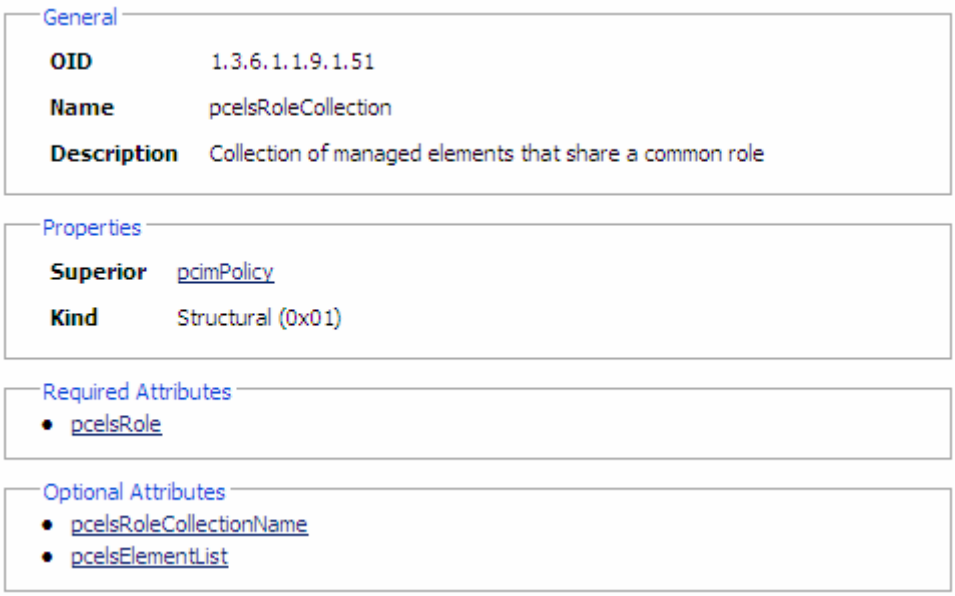


Figura 3.30 – Classe pcelsRoleCollection

Para isso esta referência aponta para um conjunto de *roles* existentes sobre um repositório interno que agrupa objectos do directório com as mesmas *roles* através da classe pcelsRoleCollection, como se poderá constatar pela Figura 3.30.

3.4.10 Variáveis temporais

A informação temporal deverá ser oriunda de uma fonte fidedigna, geralmente um servidor NTP a funcionar fiavelmente na rede, acessível directamente ou através de uma entidade "Monitor" que manipule todas as necessidades relativas a variáveis temporais.

Os objectos sobre os quais a plataforma deve permitir a criação e gestão de políticas são os indicados no RFC 3060 relativamente à dimensão tempo e indicadas de forma simples de seguida.

- Hora
- Dia da semana
- Dia do mês
- Mês do ano
- Duração

Para lá destas simples representações que de alguma forma têm de ser usadas na construção das condições nas políticas, temos ainda a necessidade de criar esquemas específicos para representar periodicidades complexas ou validações de acções apenas em determinados dias da semana, do mês ou do ano.

O fuso horário é também uma variável a ter em conta, e tratável através do atributo pcimTPClocalOrUtcTime.

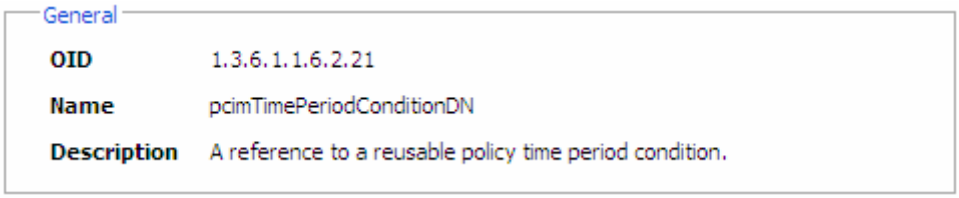
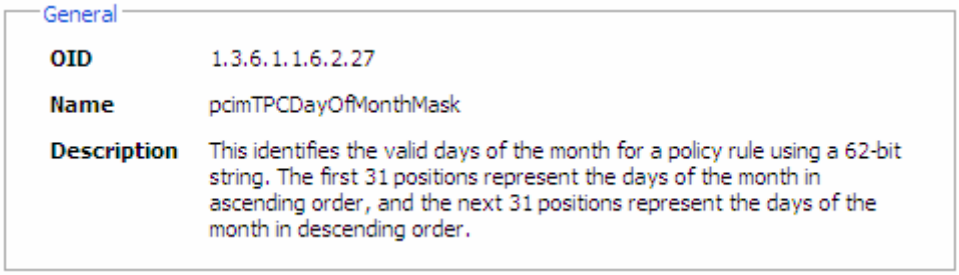


Figura 3.31 – Atributo pcimTimePeriodConditionDN

Os seguintes atributos podem ser usados inclusivamente por referenciação de condições já criadas, usando por exemplo o atributo pcimTimePeriodConditionDN da Figura 3.31. A Figura 3.32 descreve os atributos indicados nos modelos PCIM e PCIME.



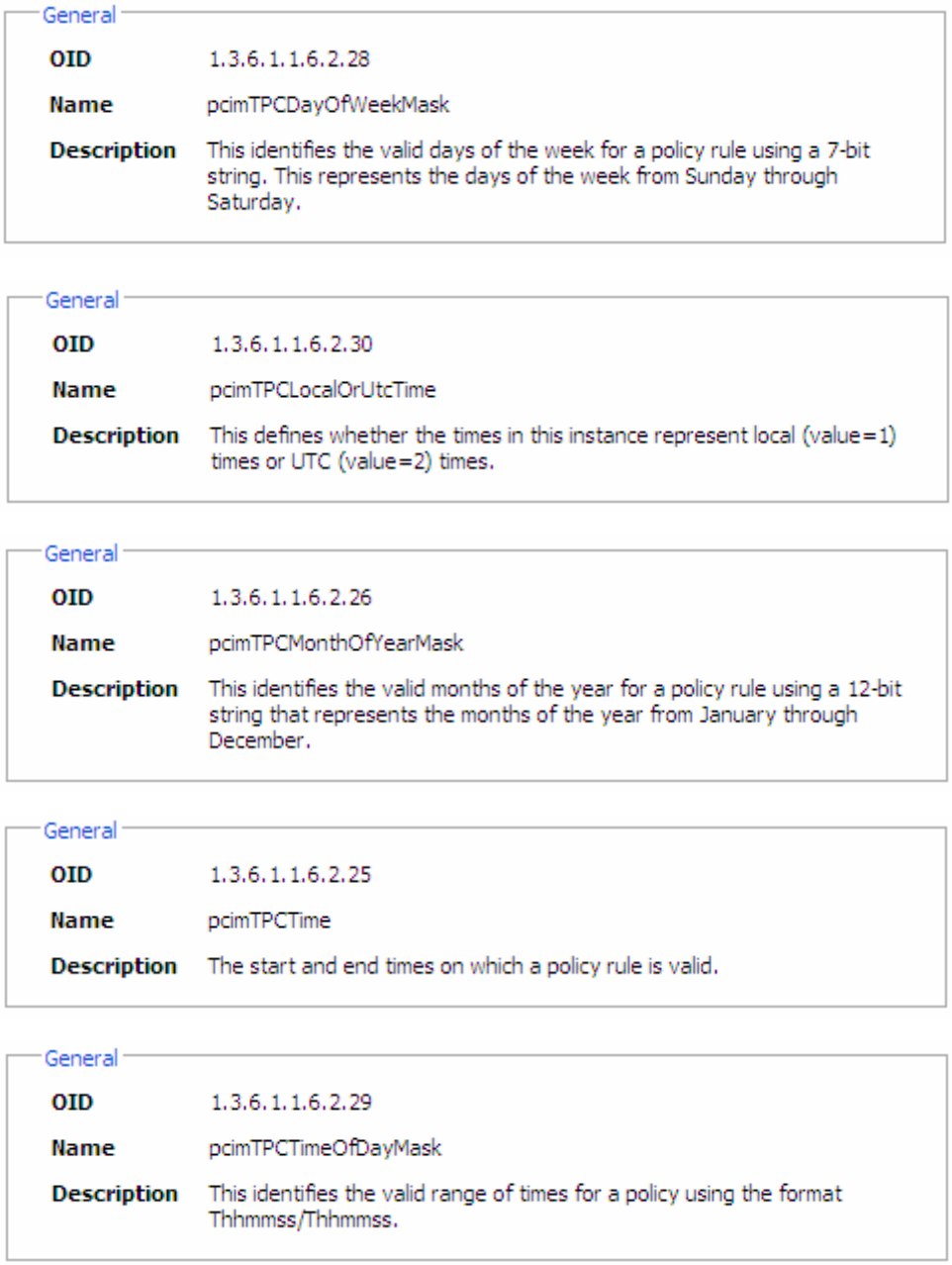


Figura 3.32 – Atributos para tratar valores temporais.

3.4.11 Variáveis inerentes aos equipamentos terminais

Há ainda um conjunto de informações que fazem parte do modelo e que não residem no repositório, mas nos próprios equipamentos terminais. Este bloco de informação define o conjunto de variáveis que podem ser consultadas e averiguadas localmente nos PEPs. Todo o conjunto de informação existente sob por exemplo a base MIB nos *router*s, é candidato a pertencer a este conjunto.

Imaginemos com exemplo uma política que altera as prioridades do tráfego caso as principais interfaces de um conjunto de *router*s mostrem sinais de congestionamento.

Aqui o teste à interface para saber do seu estado de congestionamento, pode ser testado com base na consulta no MIB do valor respectivo desde que sejam dadas instruções ao *router* para averiguar essa variável localmente e de que forma. A aplicação de meta-políticas [80] refere-se a esta funcionalidade, e através dela é possível passar muita da "inteligência" para os nós da rede, permitindo-lhes analisar a validade das condições e aplicar as acções correspondentes sem necessidade de contactar o Servidor de Políticas.

A implementação das capacidades necessárias para tratar meta-políticas, passa por implementar um método construtor e interpretador de outro tipo de PIB, o qual terá uma estrutura adequada para o efeito incluindo nomeadamente os métodos de averiguação de variáveis (local ou via PDP) [80]. Para lá desse método, há ainda a possibilidade de implementar outros métodos via *scripting*, CLI, para colher os valores para as variáveis locais necessárias. O directório guardará neste caso a informação das variáveis e valores relativos a todas essas entidades e guardá-los-à nos repositórios internos para reutilização, depois dos processos de configuração das políticas adequadas terem ocorrido ao nível da aplicação.

3.4.12 Quadro resumo

A título de resumo é apresentado um resumo em forma de quadro na Tabela 3.5 das principais classes usadas em cada uma das entidades organizacionais tratadas neste Capítulo e os correspodentes *schemas* onde se encontram definidas as classes.

Entidades	**Localização**	**Classes Usadas**	***Schemas* OpenLDAP**
Utilizador	Repositório LDAP	person inetOrgPerson posixAccount ipHost	core.*schema* inetorgperson.*schema* nis.*schema* nis.*schema*
Departamentos	Repositório LDAP	organizationalUnit	core.*schema*
Localizações físicas	Repositório LDAP	locality pcelsIPv4AddrValueAuxClass ipNetwork	core.*schema* cordenapcime.*schema* nis.*schema*
Ambientes aplicacionais	Repositório LDAP	applicationEntity[2] ipHost ipService	core.*schema* nis.*schema* nis.*schema*
Processos de negócio	Repositório LDAP	applicationProcess pcelsStringValueAuxClass	core.*schema* cordenapcime.*schema*

[2] o atributo presentationAddress por questões de simplificação foi passado a opcional.

		pcelsRoleCollection (*roles* por processo de negócio)	cordenapcime.*schema*
Referenciação temporal	Servidor NTP directo ou via Monitor no Servidor de Políticas	Atributos: pcimTimePeriodConditionDN pcimTCPDayOfWeekMask pcimTCPLocalOrUtcTime pcimTCPMonthOfYearMask pcimTPCTimeOfDayMask pcimTPCTime	cordenapcim.*schema*
Variáveis dos Equipamentos Terminais	Equipamentos Terminais	MIB Específicos	--
Políticas	Repositório LDAP	Classes definidas pelo Policy Common Information Model e PCIM Extensions nos RFC 3703 e mais recentemente o RFC 4104 respectivamente. Os objectos de topo derivam das classes do DMTF C.I.M. versão 2.5	cordenacim25.*schema* cordenapcim.*schema* cordenapcime.*schema*

Tabela 3.5 – Tabela resumo das classes usadas

Os *schema*s indicados podem ser consultados no seu detalhe e especificação nos respectivos Anexos.

3.4.13 Inteligência de rede

Entende-se com o modelo proposto potenciar o cumprimento de dois objectivos primordiais da Gestão de Redes. Assim para uma qualquer estrutura de rede, heterogénea ou não, é de primeira importância a necessidade de:

- disponibilizar os serviços definidos necessários e suficientes às aplicações e utilizadores;
- usar para isso os recursos disponíveis da forma mais eficiente.

Estes dois objectivos são obviamente a base de uma gestão eficiente. Apesar da sua simplicidade, garantem a clarificação das metas que devemos perseguir. Neste sentido, pretende-se antes de mais o contributo da rede ao nível do SLA definido e do seu correspondente SLS, ou ainda de uma forma mais informal, que simplesmente garanta o

funcionamento de uma determinada aplicação de negócio de acordo com os requisitos definidos.

Por outro lado, é importante salvaguardar que são mobilizados os recursos da forma mais eficiente possível, rentabilizando o seu uso, e garantindo sempre o nível de serviço pretendido pela entidade cliente.

PBNM serve neste contexto como a base para implementar os dois objectivos mencionados de uma forma inovadora porque permite gerir os recursos sem nunca perder estes objectivos de vista, recorrendo também a todo um leque de políticas, regras e acções. Estas terão que estar correctamente definidas de acordo com os objectivos identificados ou acordados.

No entanto, constata-se que existem alguns pontos merecedores de atenção, nomeadamente no que respeita à escalabilidade [89], desempenho e capacidade de resposta a eventos inesperados, entre outros pontos [86].

Face a estas limitações, o presente trabalho defende um modelo baseado na delegação de funções para os elementos inferiores da arquitectura PBNM, como forma de diminuir o tempo de resposta na aplicação das políticas à rede e de tornar tais componentes chave da arquitectura mais autónomos, retirando daí também uma diminuição de alguns riscos inerentes à diminuição da dependência das comunicações com os PDPs, repositórios LDAP e melhorias na utilização eficiente da rede.

Ao autonomizar mais os PEPs, pretende-se capacitá-los não apenas com acções, mas também com regras, tal como defendido nos trabalhos sobre meta-políticas [80], aumentando o espectro de acção destes pontos da rede, e diminuindo em grande medida as necessidades de controlo via PDP das decisões a aplicar. Estas podem ser fruto eventualmente de funções de monitorização existentes nos próprios PEPs.

Os PDPs serão retirados do intercâmbio com os PEPs nos casos em que estes não apresentam valor acrescentado ao processo. Nesta arquitectura, a função dos PDPs fica assim, mais focada nas funções de *Policy proxy*, nomeadamente (i) compatibilizando políticas com equipamentos que não apresentam suporte nativo a protocolos de controlo de políticas (e.g. COPS-PR); (ii) na resolução de conflitos aquando da aplicação de políticas divergentes, e finalmente (iii) na função de integração das funções de monitorização e controlo da rede.

Como em qualquer hierarquia funcional, a função de delegação traz consigo a necessidade da notificação periódica. Nesta abordagem a função de notificação é indexada à periodicidade de controlo operacional nos períodos mínimos definidos no SLA ou SLS, adicionando a essa métrica o tempo de resposta necessário para a correcção de eventuais desvios.

Não deixa de ser interessante neste momento, fazer uma breve comparação com a gestão de equipas no nosso Mundo real. Tendo como base os PEPs como membros de uma equipa que executa as tarefas planeadas pelo seu coordenador – o PDP.

Vejamos brevemente onde nos leva tal comparação:

Tal coordenador terá como missão executar com a sua equipa os objectivos que lhe foram propostos (SLA), garantindo sempre que os diversos membros da mesma, executam as tarefas que lhes são atribuídas dentro dos prazos acordados (SLS) e com os recursos que têm disponíveis para o efeito (REDE), alocando a quantidade de recursos necessária e suficiente (QoS). No caso de existir um excesso de recursos, estes poderão sempre ser canalizados para outros serviços que têm de ser da mesma forma garantidos (outras aplicações ou SLA sobre a mesma estrutura), ou então servem simplesmente como alternativa para o caso de indisponibilidade de alguns dos membros principais (*Backup*s, redundância), embora possam ser usados para outros fins durante os períodos em que não são necessários - *overprovisioning.*

De seguida, pensando um pouco no processo de distribuição de funções, repare-se no modelo de delegação. Inicialmente, o coordenador atribui determinada acção ao seu colaborador (REQ msg). Tal acção pressupõe o cumprimento com sucesso do trabalho executado, momento em que o resultado final da acção deve ser reportado ao seu coordenador (REP msg). Por outro lado, objectivos que se estendem no tempo, como por exemplo, a manutenção de um orçamento anual (SLS; *Policy*), implicam mais do que um único relatório no final do período, nomeadamente uma periodicidade mensal para verificar desvios e accionar correcções enquanto é tempo. Por outro lado, um acompanhamento diário neste caso parecerá certamente exagerado, a tal ponto que poderá inclusivamente diminuir a capacidade dos recursos em executar as tarefas que lhe foram confiadas, por diminuir o tempo disponível para a execução das suas funções – sobrecarga de recursos.

A tendência para capacitar cada vez mais os elementos da equipa a agirem autonomamente e de forma correcta, é importante no sentido em que potencia o desempenho do elemento e o seu conhecimento para ultrapassar situações novas e não previstas, quer em capacidade e rapidez de resposta - resiliência da rede.

Situando obviamente cada abordagem dentro do seu próprio Mundo, não podemos deixar de reparar nas semelhanças e no que pode ser transferido para o modelo PBMN da experiência do quotidiano que vivemos.

Adicionalmente, os elementos da arquitectura descritos na secção anterior, a sua existência puramente material e a sua repetibilidade de conduta, em certa medida potenciam esta abordagem de delegação e auto-suficiência, desde que a robustez e fiabilidade das partes constituintes esteja garantida. Em resumo, pretende-se apenas com esta breve comparação, ilustrar as razões de base do modelo proposto para a relação dos elementos de rede desta plataforma.

De facto entende-se que face às exigências actuais nas redes de comunicações, nomeadamente em termos de desempenho, capacidade de encaminhamento, instantaneidade das decisões no seio da rede, etc, as abordagens com real capacidade para responder a tais requisitos serão as baseadas em mecanismos de *Provisioning*.

As alternativas baseadas em mecanismos de *Outsourcing*, ou seja por envolvência directa do PDP nas decisões, embora funcionalmente se entenda fazer sentido, o seu peso e medida deverá ser bastante criterioso principalmente por questões de desempenho. É imperativo que face às necessidades indicadas, os mecanismos de rede sejam capazes de gerir com o mínimo tempo decorrido as decisões de marcação de pacotes, encaminhamento, policiamento, etc.

Este ponto leva-nos a uma constatação nuclear deste trabalho, que aposta na autosuficiência dos equipamentos de rede, e nas iniciativas para aumentar a sua "inteligência" de rede. A Tabela 3.6 resume as comparações entre a gestão das equipas e a abordagem em PBNM.

Gestão RH's	PBNM
Coordenador	PDP
Objectivo a cumprir	SLA
Colaboradores	PEPs
Recursos	Rede
Gestão de Recursos	QoS, *Overprovisioning*
Autonomia	Via *Policy Provisioning* - Decisões tomadas no PEP de acordo com as políticas recebidas.
Relatórios	COPS REP msg
Capacidade de resolução de situações novas	Resiliência
Experiência	Acções via uso de métodos de IA, *Data Mining* ou consulta de histórico
Tarefas atribuídas	COPS REQ msg
Gestão de Conflitos	Arbitragem no PDP, ou gestão por prioridades
Periodicidade de envio de relatórios	Definido nas tarefas específicas, ou de acordo com o SLS, SLA ou preferências de gestão.

Tabela 3.6 – Comparação entre a Gestão de RH's e PBNM.

Esta abordagem conjunta define assim uma linha de pensamento do que se entende que deve ser a PBNM, potenciando a delegação para os PEPs das políticas, suas con-

dições, regras e acções, de forma a melhorar a qualidade do serviço prestado pela rede, e optimizando a utilização dos recursos disponíveis.

Integrando a visão aqui descrita com a plataforma no seu todo desde o modelo de informação descrito, o fluxo de processos e operações, e os mecanismos de tradução apresentados, obtemos a plataforma CORDENA no seu todo.

3.4.14 Visão Geral da Arquitectura

Apresenta-se de seguida na Figura 3.33, o Diagrama Geral da Plataforma CORDENA contemplando os blocos mais importantes e retratados ao longo das últimas secções.

Façamos um breve resumo de um processo de selecção e aplicação de políticas na generalidade. De notar que o processo pode ser despoletado por diversas formas: (i) por arranque de um elemento da rede, caso em que ele se anuncia ao PDP, indica as suas capacidades e fica à espera de mensagens de decisões (DEC); (ii) por actualização das políticas já aplicadas, caso em que o *triggering* tem de existir para despoletar as necessárias actualizações eventualmente nos dois sentidos (para o repositório, e para os PEPs), entre outras.

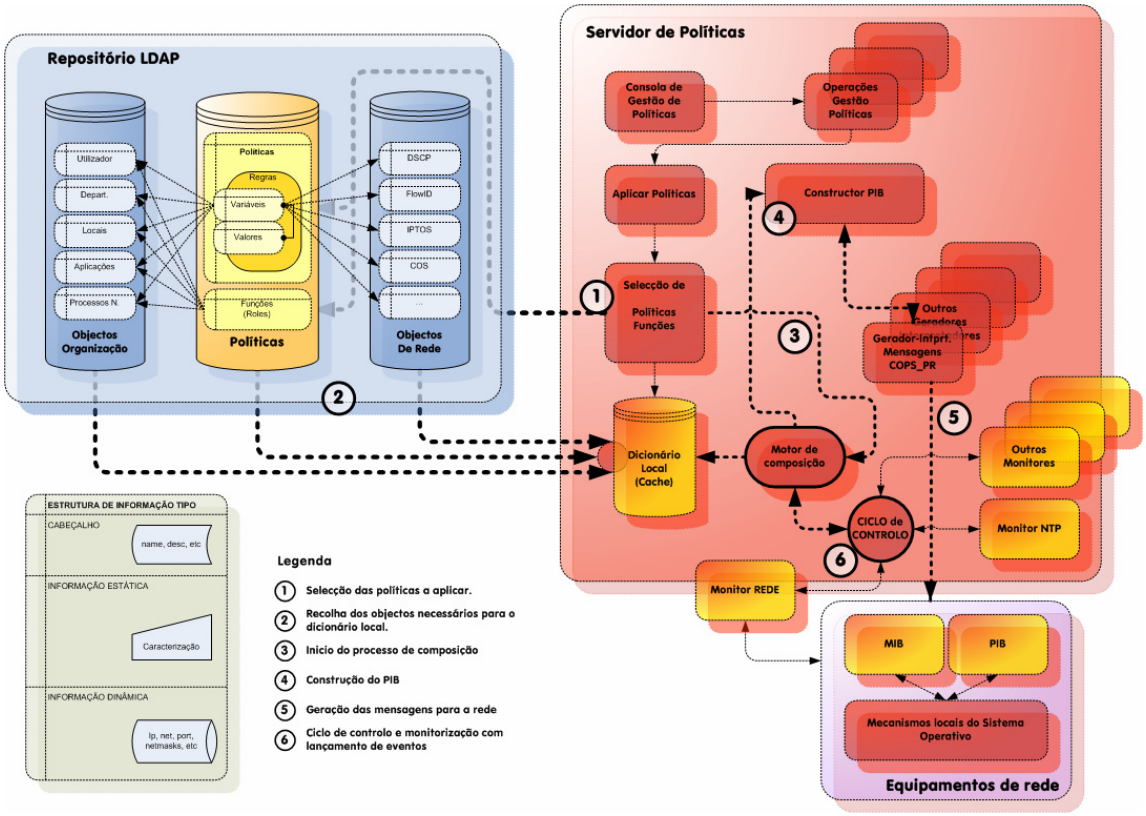


Figura 3.33 – Visão Geral da arquitectura CORDENA.

Nada melhor que seguir o processo de exemplo indicado na Figura 3.33. Uma cópia aumentada é apresentada em anexo para facilitar a leitura.

Através de um qualquer processo de gestão das políticas, com o auxílio de uma linguagem de especificação ou de versão gráfica numa consola com as funcionalidades adequadas, são definidas as políticas a aplicar à rede. Esta aplicação pode ocorrer directamente sobre os objectos pretendidos, ou de um forma agregada atribuindo as políticas às *roles* adequadas.

De seguida todas as variáveis e valores necessários são descarregados para uma instância de base de dados local (que pode ser relacional) que guarda além de informações de estado do próprio servidor de políticas, as informações descarregadas do repositório.

Esta funcionalidade ganha contornos importantes na possibilidade de distribuir a arquitectura em servidores físicos afastados na estrutura de rede. Caso essa situação seja crítica, devem ser implementados mecanismos de replicação do directório para instâncias mais próximas do servidor de políticas.

Com as partes constituintes disponíveis e traduzidas, despoleta-se o processo de composição de acordo com os *namespaces* definidos nas estruturas de dados da rede, no exemplo o PIB para Serviços Diferenciados [16], ou o PIB alterado para alocar meta-políticas à rede [80].

Os métodos construtores vão invocar os objectos do dicionário local, construir tais estruturas de dados e posteriormente gerar as mensagens protocolares com a codificação definida. Estas operações, apesar de conceptualmente simples, podem encerrar alguma complexidade mecânica, que pode ser objecto de estudo na optimização algorítmica como forma de gerar os pacotes com a máxima eficiência. Finalmente os pacotes são constituídos e enviados à rede.

Entretanto há algumas entidades bastante importantes que se salientam uma vez mais: os monitores que monitorizam o estado da rede para colher as necessárias quantificações de tempo; o estado do domínio em termos de congestão, etc; assim como o "batimento cardíaco" do sistema - o Ciclo de Controlo que representa um processo que varre o estado de todos os monitores, analisa os pedidos e despoleta as necessárias acções.

É um processo de controlo que dada a sua importância pode ter na sua génese soluções de tolerância a falhas, de modo a ser mais resiliente na protecção contra falhas nomeadamente as arbitrárias ou bizantinas.

Finaliza-se assim a apresentação do detalhe arquitectural da Plataforma CORDENA e dos seus blocos constituintes. De seguida far-se-á uma breve passagem pelas aplicações possíveis.

3.4.15 Notas Finais

Está descrita a arquitectura CORDENA e as suas partes constituintes.

As secções anteriores fizeram uma descrição de todos os processos envolvidos e das considerações mais importantes passando em ordem os principais passos no processo de tradução de políticas e salientando que alguns dos processos de tradução podem ocorrer antes de serem realmente necessários, podendo dessa forma evitar os problemas de escalabilidade e complexidade associados. Para isso, o modelo apresentado defendeu entre outras abordagens que alguns elementos podem ser traduzidos ao nível do directório.

Foi ainda apresentado um modelo Humanizado da rede como forma de ultrapassar algumas limitações no modelo em si, nomeadamente as questões de escalabilidade.

Em termos da tradução para os elementos da rede (PEPs), foi usado um exemplo baseado no protocolo COPS-PR.

Convém salientar, tal como detalhado nos Módulos de Construção de Mensagens no Servidor de Políticas, que outros protocolos poderão ser utilizados, bastando para isso enquadrar na arquitectura os blocos constituintes necessários, sempre numa perspectiva modular sem alterar o núcleo da plataforma.

Exemplos de outros protocolos podem incluir CORBA, ou mais recentemente NSIS, uma plataforma nova de sinalização que pode também ser usada para transporte de políticas. Os estudos à volta do NSIS congregam-se no grupo de trabalho *IETF - Next Steps in Signaling (NSIS)*.

3.5 Perspectivas e Exemplos de Aplicação

Por mais simples que pareça, o passo necessário para a divulgação e implementação generalizada de uma ideia como esta (PBNM) passa apenas pela adopção. Adopção por parte dos diversos fabricantes, fornecedores de soluções e serviços de forma a alargar o parque de equipamentos que suporte nativamente as tecnologias envolvidas.

É claro que esta visão simplista do problema põe de parte toda a complexidade das relações comerciais, estratégicas e económicas, bem centradas nestas temáticas e que têm um impacto representativo na evolução e adopção das tecnologias, muitas vezes contra aquilo que por motivos puramente científicos seria de esperar.

Assim, um dos pontos que esta plataforma pretende expor é a simplicidade relativa com que uma tecnologia como esta pode ser mapeada sob uma organização nas suas várias

componentes técnicas. Vejamos por exemplo alguns casos de necessidades e problemas que poderão ser complementados com PBNM dinamicamente:

- **Exemplo 1**: Gestão de um acesso condicionado à Internet durante horas de expediente.
- **Exemplo 2**: Aumento da prioridade de tráfego aplicacional nos últimos dias do mês, devido a operações intensivas típicas de fecho de mês.
- **Exemplo 3**: Diferenciação da prioridade do tráfego Internet para acesso a soluções de *Internet Banking, e-business* e *e-procurement.*
- **Exemplo 4**: Aumentar a percentagem de largura de banda disponível em vários *links* internacionais simultaneamente durante 2 horas para suportar uma videoconferência multiponto.
- **Exemplo 5**: Dar máxima prioridade para a publicação de relatórios oriundos de operações intensivas de *Datawarehouse* durante o período nocturno de modo a garantir a sua publicação às primeiras horas da manhã.
- **Exemplo 6**: Usar aplicações secundárias até ao limite sob o qual as aplicações críticas de negócio não são afectadas, nomeadamente *downloads*, transferências via FTP ou instalação a pedido de software.
- **Exemplo 7**: Atribuir limites de largura de banda, ou quota de tráfego diário por utilizador de acordo com as suas funções, para o centro de dados, Internet, etc.
- **Exemplo 8**: Proibir explicitamente o acesso de grupos de utilizadores a determinados serviços ou servidores, nomeadamente para canalizar acessos para mecanismos de balanceamento de carga, ou simplesmente por motivos de acesso condicionado a manutenção de políticas de acesso.
- **Exemplo 9**: Desviar instantaneamente acesso de grandes grupos de utilizadores a diferentes nós de clusters, a diferentes troços de rede, por motivos de manutenções programadas, evitando o necessário *downtime* a utilizadores e acima de tudo de uma forma totalmente transparente para eles, evitando eventualmente avisos prévios de indisponibilidade de sistema.
- **Exemplo 10**: Gerir o acesso a servidores de desenvolvimento e teste permitindo sob os mesmos recursos programar acções de formação e testes em massa, em detrimento de acções de desenvolvimento e vice-versa, de acordo com as necessidades evitando neste caso o dobro do investimento.
- **Exemplo 11**: Análise técnica da própria rede de um fornecedor de serviço, para apurar que nível de serviço (SLA) está apto a forneceder aos seus clientes. [79]

Muitos exemplos haveria para explorar que ilustram a utilidade da Gestão de Redes baseada em Políticas, contudo este conjunto elucida bem as possibilidades de exploração da tecnologia.

Os próximos pontos tecerão ainda algumas considerações acerca das potenciais vantagens do uso de Gestão baseada em Políticas aplicada ao controlo de SLAs, na tolerância a falhas, na resiliência da rede e na gestão de conflitos.

3.5.1 Definição e Controlo de SLA / SLS

Num exemplo indicado em [79], o autor faz referência a um exemplo de política que verifica simplesmente quais os serviços que o fornecedor de conectividade está capacitado para dar como base num conjunto de valores de um SLA, nomeadamente latência, tempo de resposta, etc. Fazer um exercício destes com recurso a políticas é um excelente exemplo das capacidades da plataforma.

Na verdade, uma das maiores vantagens na aplicação destas plataformas à Definição e Gestão de SLAs é o facto de disponibilizar informação de qualidade no que respeita a valores de monitorização, métricas, SLAs, etc, permitindo dessa forma a tomada de decisões conscientes e estrategicamente correctas pois os valores e as métricas existem.

Pode parecer desenquadrado, mas em muitos casos práticos, apesar dos serviços estarem em produção e nos clientes (ex. QoS), não há evidência do seu impacto e de quais os resultados, sendo o único resultado a certeza de que tudo está devidamente configurado. De qualquer forma esta resposta não é suficiente!

Gerir a rede desta forma permite entre outras vantagens:

- Saber claramente o estado da rede.
- Analisar adequabilidade dos pedidos à rede de forma quase instantânea.
- Mensurar as principais variáveis da rede em termos de caracterização.
- Garantir informação de qualidade
- Descentralizar o controlo e verificação de SLA/SLS.

De uma forma geral capacita o gestor com informação de qualidade medida no seio da infra-estrutura e não por aproximação, ou seja, dá ao gestor as capacidades analíticas necessárias para uma tomada de decisão criteriosa e adequada.

3.5.2 Redundância e Tolerância a Falhas

Antes de mais é importante clarificar que por si só, as infra-estruturas não vão ganhar mais tolerância a faltas pelo simples facto de serem geridas de outra forma.

De facto, se um determinado servidor crítico está totalmente só sem qualquer medida de contingência associada, é óbvio que usar PBNM integrando essa entidade em várias políticas adequadas apenas nos dá capacidade e flexibilidade de gestão. Mas é justamente aqui que surgem algumas vantagens muito interessantes.

Hoje em dia, face a problemas graves numa parte de uma qualquer infra-estrutura de rede, em primeiro lugar muito raramente os utilizadores finais não se apercebem dos problemas e da quebra de serviço, e em segundo lugar, dependendo da gravidade da situação e dos planos de contingência existentes, as necessidades de recursos num curto espaço de tempo são imensas! Imagine-se o mesmo cenário, mas com recurso a PBNM com uma plataforma como a que aqui foi apresentada.

Dependendo das áreas afectadas em caso de desastre, nomeadamente da capacidade da própria plataforma de gestão e das redundâncias dela própria (p. e. com PDPs alternativos em utilização) fazer encaminhamento imediato a outras instâncias aplicacionais, atribuindo alta prioridade a grupos completos de utilizadores, pode ser totalmente transparente, adequar uma percentagem de diminuição de serviço pelo uso de linhas de comunicação alternativas com menor capacidade, é outro exemplo.

Muitos mais exemplos poderiam ser citados a propósito da aplicação desta tecnologia. Para o efeito penso que serão já suficientes, pois o que se pretende ilustrar é justamente esta capacidade de adequar os meios e recursos existentes às situações e necessidades quotidianas ou não de uma típica infra-estrutura de qualquer dimensão.

É aí que reside uma das grandes vantagens desta tecnologia, na melhoria significativa da capacidade operacional, ao permitir diminuir os tempos de resposta, ao tornar os problemas transparentes para os clientes internos ou externos, enfim ao permitir à entidade gestora dar um serviço adequado e de qualidade.

3.5.3 Resiliência

A resiliência é um caso particular da redundância e tolerância a falhas. Entende-se neste contexto a noção de resiliência como mais do que a simples tolerância a falhas.

A redundância e tolerância a falhas debruçam-se na maior parte das vezes sobre falhas previstas e conhecidas. As soluções encontradas ao longo dos últimos tempos reflectem sempre uma resposta a um problema conhecido, exceptuando alguns casos, como por exemplo um *cluster* de servidores, o qual acaba por tolerar qualquer problema nos servidores, desde que não sejam problemas com os mecanismos que implementam a tolerância.

Por outro lado resiliência diz respeito a preparar as aplicações e as infra-estruturas a tolerar problemas impossíveis de prever e sem à partida um perfil conhecido. Como?

Esta simples questão de difícil resposta, cria um desafio ao qual se responde desde já com algumas ideias:

- **Através da implementação de algumas medidas preventivas simples**, buscando a simplicidade na definição das soluções alternativas. Por vezes diminuir à complexidade das soluções é a melhor forma de garantir um serviço mínimo em caso de falhas. Assim poder-se-á evitar mais pontos de falha introduzidos com a própria complexidade das soluções alternativas, que muitas das vezes são elas próprias o cerne dos problemas, inviabilizando a funcionalidade prevista das soluções.
- **Através da implementação de medidas curativas eficazes**, com base nos efeitos sobre a infra-estrutura, ou seja, com base em algumas características e no estado da infra-estrutura o sistema age por iniciativa própria tomando medidas de contingência por si e automaticamente. Parece algo longínquo mas não o será com recurso a uma plataforma como a CORDENA. De entre as suas peças constituintes, falou-se dos Monitores, do Ciclo de Controlo e dos Blocos Geradores de mensagens e políticas. Um qualquer incidente que altere a estabilidade do estado da rede, traduzido por exemplo num conjunto de métricas definidas como um SLA, cria necessariamente alterações a essas métricas. De "olhos vendados" o sistema terá toda a capacidade para detectar onde está o problema e quais as alternativas, sejam elas opções de encaminhamento de tráfego por rotas alternativas, ou de uma forma generalizada, passar a um estado por defeito previamente definido, activando-o em caso de determinadas falhas.

Muito mais haveria que falar neste ponto, de qualquer modo fica a ideia geral de que o objectivo primeiro é de facto capacitar os sistemas e a gestão de infra-estruturas, redes e não só, para serem auto-suficientes e se auto adaptarem a alterações previstas ou não. Melhor do que um humano que opere face a alterações significativas, a infra-estrutura com os mecanismos de gestão aqui descritos e com toda a inteligência mecânica montada sob uma plataforma como a CORDENA, é a entidade que melhor pode gerir tais situações.

É nesta abordagem que o modelo *Provisioning* se enquadra perfeitamente ao defender alguma auto-suficiência dos elementos da rede, como vimos.

3.5.4 Gestão de Políticas Contraditórias e Outros Conflitos

Para lá do epicentro da tecnologia aqui descrita, muitas oportunidades poderão surgir após a sua génese, e no seu desenvolvimento. Embora numa perspectiva provavelmente mais visionária do que real, a gestão de conflitos e a gestão de políticas incompatíveis pode recorrer às Ciências da Computação Pericial, ou a Técnicas como as da Extracção de Conhecimento de Bases de Dados.

No momento actual e nos modelos de informação descritos, a funcionalidade que nos permite gerir conflitos de políticas é a criação de graus de prioridade atribuídos às políticas ou a grupos de políticas, assim como classificar esses grupos com alguma informação de estratégias de aplicação.

Estas funcionalidades embora possam resolver alguns conflitos específicos, carecem de mecanismos mais avançados para por exemplo tomar a medida adequada quando uma política pretende fazer uma reserva que pode por em causa outros SLA já efectivados. Na forma normal, é usada a rejeição pura e simples, uma medida limitada a um valor de prioridade e não ao estado completo da rede.

Neste ponto, em [79] aponta-se como possível solução o lançamento de politicas dinâmicas para a resolução de problemas deste tipo, assim como para um módulo específico dentro da arquitectura para a resolução de conflitos, tal como por exemplo nos estudos sobre Redes Activas retratado em [78].

De uma forma ou de outra, garantir inteligência à rede passa sempre pela existência de objectivos claros a cumprir (SLA/SLS) e pelas políticas globais que se pretende seguir no domínio de gestão.

Em caso de conflito, haverá que descobrir em primeiro lugar se se trata de um “falso conflito”, ou seja, um conflito que para o utilizador é um problema, mas que a rede pode resolver com recurso às políticas globais definidas.

Caso se trate de um conflito confirmado, entre as soluções possíveis poderemos ter a divisão de recursos, a negação com base na prioridade, ou mais ainda o recurso a motores periciais e algoritmia de extracção de conhecimento agindo de acordo com o comportamento habitual da rede, ou idealmente de acordo com as necessidades esperadas dos clientes de tais recursos.

Apenas para dar um exemplo, em caso de conflito confirmado, a solução pode passar pela não alocação de recursos, ou por alocar parcialmente os pedidos na medida da necessidade gradual, ou ainda pela recuperação de recursos alocados a outros pedidos, por haver conhecimento estatístico ou pericial de que apesar de determinados fluxos estarem alocados, não vão ser usados na sua totalidade.

Nestes casos, subsistem as questões de escolha sobre quais podem ser recuperados ou não. Com o auxílio de sistemas baseados em técnicas de inteligência artificial, estas decisões podem ser facilmente suportadas. Por outras palavras falamos de técnicas de *overprovisioning* ajustadas a gestão dos conflitos.

4 Incursões experimentais

É inegável a utilidade da experimentação para por à prova as ideias expostas. Além de se testar e comprovar a viabilidade do modelo proposto, enriquece-se o trabalho pelos problemas que emergem apenas no laboratório de ensaios, longe do papel e das definições.

Este Capítulo descreve os pontos mais importantes do trabalho que ao longo de meses se misturou entre os papéis, os esboços e determinados processos a correr num conjunto de máquinas virtuais onde foram testadas as tecnologias.

É também exposta uma proposta de implementação nas suas componentes de planeamento e especificação.

4.1 Âmbito

Neste caso específico as incursões experimentais cingiram-se aos trabalhos necessários para o controlo e apuro técnico das tecnologias envolvidas nos domínios em causa. Para isso foi necessário criar um laboratório de ensaios plural composto por várias tecnologias diferentes que vale a pena mencionar.

Das partes experimentais a que se revelou de uso mais intenso foi naturalmente o servidor LDAP, pelas sucessivas experimentações executadas sobre ele na criação da organização de base e de toda a experimentação sobre os *schemas* e os objectos testados.

É ainda feita uma menção às principais tecnologias em foco nestes trabalhos e as particularidades mais interessantes das escolhas efectuadas. Juntamente com estes esboços de investigação e teste, é proposto um modelo de planeamento do projecto de implementação e tecidas algumas considerações a propósito.

4.2 Requisitos do laboratório de testes

A peça chave das incursões experimentais deste trabalho é o laboratório de testes usado. Uma vez que o objectivo nuclear do laboratório foi testar componentes de um Sistema de Gestão de Redes baseado em Políticas numa organização entendeu-se como necessária uma determinada base técnica que representasse as infra-estruturas da organização Vnet.lab.

Esta é alias uma abordagem a ter em conta nas operações generalizadas de teste e desenvolvimento de Gestão baseada em Políticas, permitindo prever quais os comportamentos e testar os desenvolvimentos sob um ambiente totalmente controlado.

Para lá deste importante ponto, os requisitos identificados para a criação do laboratório foram os seguintes:

- A necessidade de ter um sistema de directório que guardasse de forma unívoca os utilizadores, grupos de utilizadores e se possível restantes recursos tecnológicos de forma a gerir a autenticação nos sistemas de rede e aplicações, sempre de acordo com os respectivos níveis de acesso autorizados.
- A existência de uma qualquer rede de comunicações WAN (*Wide Area Network)* que interligasse as várias delegações.

- Cada delegação deveria ter uma infra-estrutura de rede local LAN *(Local Area Network)* responsável por conectar os postos clientes aos sistemas de gestão da empresa.
- Deveriam existir sistemas de informação para todas as áreas da empresa. (produção, logística, financeira, administrativa, etc) usados por todos os colaboradores da empresa (ERP ou não).
- A empresa necessitaria de uma ligação estável e fluente à Internet para satisfazer alguns processos de negócio críticos, nomeadamente para acesso a bancos, sistemas de aprovisionamento, etc.
- Da mesma forma deveria existir uma área reservada e de acesso externo controlado para aceder ao sistema de e-Business da empresa (*Demilitarized Zone* - DMZ).
- Justamente pela existência dos dois pontos anteriores, a existência de uma *firewall* é fortemente recomendada.

Há ainda a considerar alguns requisitos técnicos adicionais, relativos à natureza de investigação desta dissertação e não pelo perfil da organização em teste. Esses requisitos são os seguintes:

- **Uso de soluções *Open Source*:** Por permitir a exploração total das soluções em estudo e de todas as suas componentes, além de conter os custos associados ao desenvolvimento. Além disso tratam-se de sistemas em produção e com fiabilidade e robustez bem confirmada ao longo dos últimos quase quinze anos de existência.
- **Representatividade da tecnologia real:** As soluções encontradas devem reflectir as preocupações reais, os dimensionamentos aproximados e as soluções técnicas mais usadas no mercado empresarial.
- **Alta disponibilidade e segurança** para as aplicações críticas de negócio, por exemplo com soluções de alta disponibilidade para aplicações críticas nomeadamente *clustering* se possível.
- **Portabilidade do ambiente:** Um tal sistema de testes, obriga naturalmente a um conjunto de equipamentos de teste razoável, desde equipamentos terminais, *routers*, comutadores, servidores, etc. O desafio lançado como requisito técnico não crítico, é tornar tal ambiente portátil por questões de produtividade e facilidade para a execução de todo o desenvolvimento e testes.

Na globalidade este foi o conjunto de requisitos equacionado na construção de um laboratório de testes que suportou os trabalhos experimentais.

4.3 Laboratório de Testes

A infra-estrutura resultante de todas as especificações e requisitos levantados anteriormente é a apresentada na Figura 4.1. Trata-se de uma infra-estrutura totalmente virtual implementada sob VMWare GSX Server 3.2. Esta foi a forma encontrada de conseguir implementar num pequeno espaço e de forma portátil toda uma infra-estrutura de teste que se veio a demonstrar imprescindível.

De notar que a opção pela versão GSX Server deveu-se ao facto de nativamente serem implementados mecanismos de particionamento da rede virtual através de dez comutadores virtuais. Dessa forma foi relativamente fácil isolar LAN e WAN permitindo criar mecanismos de encaminhamento entre as diversas redes virtuais.

Esta escolha foi apenas possível porque não fazia parte da lista de requisitos apurar e medir o desempenho, mas sim criar uma ambiente totalmente funcional em termos técnicos e principalmente que não partisse de condicionamentos de emulação, mas sim recriasse fielmente todas os servidores e *router*s constituintes.

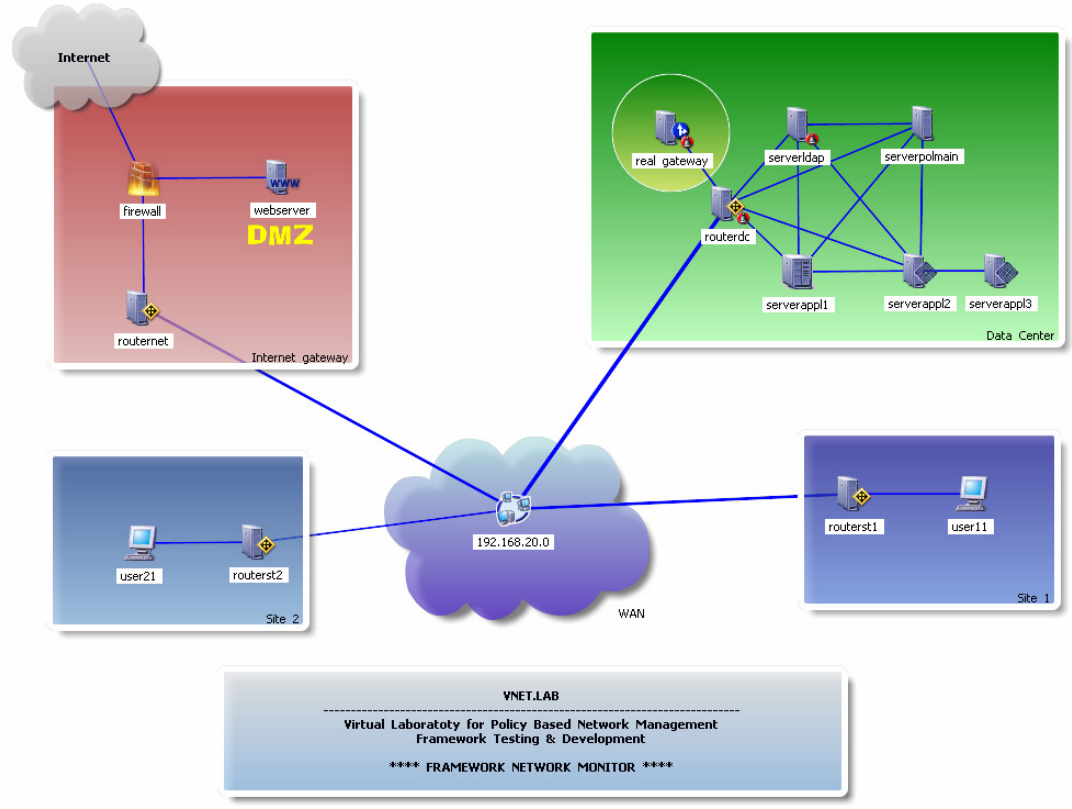


Figura 4.1 – Laboratório Virtual

A ilustração da Figura 4.1 é uma cópia do ecrã de uma ferramenta de monitorização via SNMP [133] do laboratório virtual acessível através do *router* referenciado no diagrama

como *real gateway*, o qual é a interface de rede real sob a qual existe uma *bridge* virtual de acesso ao Centro de Dados virtual.

Por motivos de direitos de autor, e da impossibilidade de desenvolver implementações sob sistemas proprietários, o que seria representativo de uma fatia dominante de mercado, são usadas componentes Open Source para as implementações dos *router*s.

Código	Nome	Função	Sistema Oper.	Aplicações	Kernel	Interface Primário (LAN)	Vmnet	RAM	Disco	CPU
R1	routerdc	Router de acesso ao data center	Linux Slack. 10.1	iproute2	2.4.26	LAN 10.10.1.1	3	16 Mb	128 Mb	Intel Pentium Mobile 1,6Mhz i686
R2	routerst1	Router de acesso ao site 1	Linux Slack. 10.1	iproute2	2.4.26	LAN 10.10.2.1	6	16 Mb	128 Mb	Intel Pentium Mobile 1,6Mhz i686
R3	routerst2	Router de acesso ao site 2	Linux Slack. 10.1	iproute2	2.4.26	LAN 10.10.3.1	7	16 Mb	128 Mb	Intel Pentium Mobile 1,6Mhz i686
R4	routernet	Router de acesso ao gateway internet	Linux Slack. 10.1	iproute2	2.4.26	LAN 10.10.4.1	8	16 Mb	128 Mb	Intel Pentium Mobile 1,6Mhz i686
Sdev1	serverdev	Servidor de desenvolvimento	Linux Slack. 10.1	--	2.4.26	LAN 10.10.1.49	3	16 Mb	128 Mb	Intel Pentium Mobile 1,6Mhz i686
S1	serverldap	Servidor de Directório e DNS Primário	Linux Slack. 10.1	OpenLDAP, Bind 9	2.4.26	LAN 10.10.1.50	3	128 Mb	1024 Mb	Intel Pentium Mobile 1,6Mhz i686
S2	serverpmt	Gestor do Framework CORDENA	Linux Slack. 10.1	MySQL + openLDAP	2.4.26	LAN 10.10.1.51	3	128 Mb	1024 Mb	Intel Pentium Mobile 1,6Mhz i686
S3	serverapp1	Servidor aplicacional simples	Linux Slack. 10.1	Geração tráfego simula	2.4.26	LAN 10.10.1.52	3	128 Mb	1024 Mb	Intel Pentium Mobile 1,6Mhz i686
S4	serverapp21	Servidor aplicacional em cluster - 1º nó	Solaris 10 Zone 1	Geração tráfego simula	5.1	LAN 10.10.1.53	3	256 Mb	6144 Mb	Intel Pentium Mobile 1,6Mhz i686
S5	serverapp22	Servidor aplicacional em cluster - 2º nó	Solaris 10 Zone 2	Geração tráfego simula	5.1	LAN 10.10.1.54	3			Intel Pentium Mobile 1,6Mhz i686
S6	serverweb	Servidor web	Linux Slack. 10.1	Apache Web Server	2.4.26	LAN 10.10.5.50	1	64 Mb	256 Mb	Intel Pentium Mobile 1,6Mhz i686
S7	serverfrw	Firewall + proxy corporativa	Linux Slack. 10.1	iptables	2.4.26	LAN 10.10.4.50	8	64 Mb	256 Mb	Intel Pentium Mobile 1,6Mhz i686
W1	wsuser11	estação de trabalho	Linux Slack. 10.1		2.4.26	LAN 10.10.2.x (via DHCP)	6	8 Mb	128 Mb	Intel Pentium Mobile 1,6Mhz i686
W2	wsuser21	estação de trabalho	Linux Slack. 10.1		2.4.26	LAN 10.10.3.x (via DHCP)	7	8 Mb	128 Mb	Intel Pentium Mobile 1,6Mhz i686
	TOTAIS							864 Mb	10624 Mb	

Tabela 4.1 – Especificações das máquinas virtuais

No entanto, apesar dessas limitações, que poderão ser reduzidas pelo uso de equipamentos que implementem COPS-PR e os mecanismos locais para traduzir políticas DiffServ nos mecanismos próprios do equipamento terminal (PEP), esta implementação pode facilmente ser estendida pela integração de vários directórios proprietários, desde que implementem a norma LDAP, como sejam por exemplo a Active Directory da Microsoft ou o eDirectory da Novell.

A Tabela 4.1 detalha as características de cada máquina virtual dentro do laboratório. O ambiente descrito encontra-se implementado sob VMware GSX Server 3.2 numa estação de trabalho Intel Pentium Mobile 1.6Mhz com 1Gb RAM e 80Gb de disco rígido sob Windows XP Professional.

Sobre esta infra-estrutura foram configurados os serviços normais de uma infra-estrutura técnica, nomeadamente o serviço de directório OpenLDAP, DNS com Bind 9, serviços de *routing* via iproute2, já incluído no *kernel* 2.4.26, entre outros.

A Figura 4.2, mostra várias janelas de onde se destaca uma janela com Linux em primeiro plano, e em segundo plano a consola de monitorização do ambiente virtual com as várias máquinas virtuais e o seu estado.

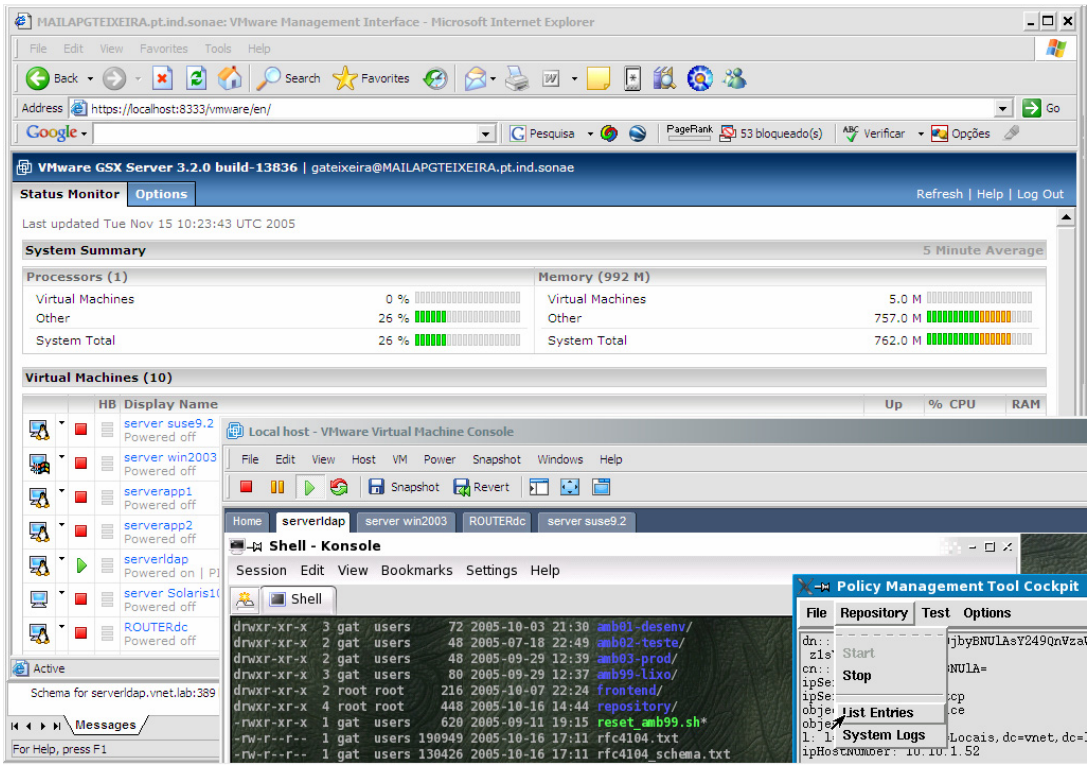


Figura 4.2 – Laboratório Virtual em uso sob VMware GSX

4.4 Tecnologias Envolvidas

Mais do que enumerar as ferramentas, métodos e tecnologias usadas do decurso desta dissertação, esta secção acrescenta em cada caso, as particularidades que unem tais tecnologias a este trabalho. Faz-se assim uma referência às peças constituintes usadas directa ou indirectamente relaccionadas com o tema principal do trabalho – PBNM.

4.4.1 Sistema Operativo Linux

Não valerá a pena detalhar todas as particularidades de um domínio já tão bem debatido e conhecido com o sistema operativo Linux. A questão principal residiu em escolher qual a melhor distribuição para os requisitos identificados, principalmente entre a Distribuição Suse 9.2 [128] e Slackware 10 [130]. Fica apenas aqui uma referência à distribuição Slackware 10 por de entre as várias existentes na comunidade ter sido a mais "limpa" de conteúdos não necessários.

Por outro lado foi também a escolhida porque embora a sua configuração tenha uma componente bastante maior de configuração manual que as restantes distribuições, o facto de ser necessário entender exactamente o que se está a passar no sistema opera-

tivo, quais os processos configurados e de que forma, tornaram esta distribuição trabalhosa, mas recompensadora. A Figura 4.3 ilustra uma normal sessão de trabalho sob Linux com várias janelas de trabalho abertas.

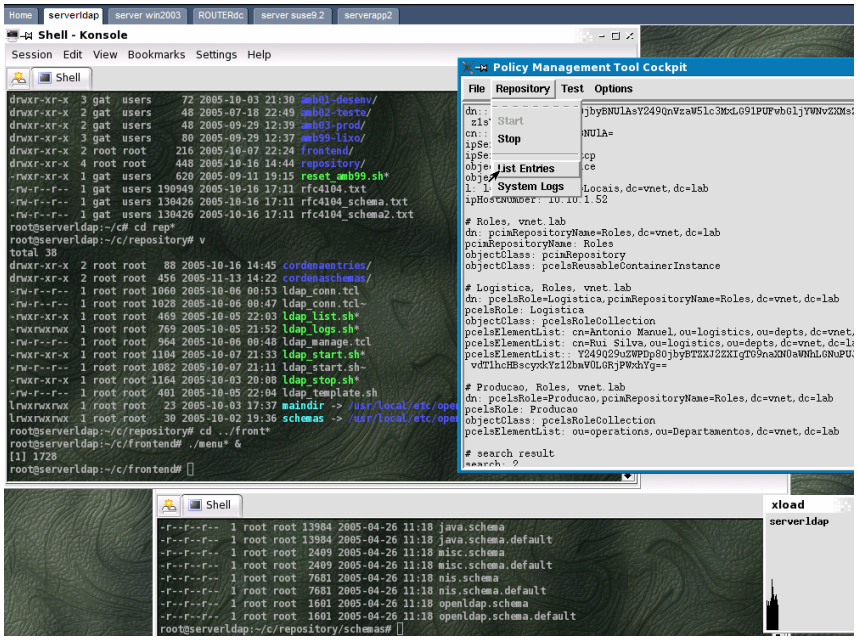


Figura 4.3 – Sessão de trabalho em Slackware Linux com o ambiente KDE

Uma das razões mais importantes porque se optou por uma distribuição limpa de funcionalidades acessórias, magra e com uma componente manual, foi o facto de ser necessário multiplicar várias vezes os servidores, *router*s, etc num laboratório virtual onde os recursos de RAM e CPU eram limitados. Só desta forma foi possível extrapolar o laboratório virtual com várias máquinas, com baixos consumos de recursos cada.

4.4.2 Virtualização de hardware

Como já se pode constatar, a bancada de trabalho usada recorreu a técnicas de virtualização como forma de permitir portabilizar um ambiente que de outra forma seria difícil de conseguir, assim como todo o equipamento físico necessário.

Após equacionar os requisitos, várias soluções de virtualização foram analisadas. Entre elas as mais importantes foram as várias versões VMware, nomeadamente a versão GSX [107] e ESX Server, a solução Microsoft Virtual Server [129], o XEN da XenSource [70] [110] e o emulador Qemu [124].

A primeira selecção foi levada a cabo pela necessidade de ser pretendido um ambiente de partilha de recursos com o maior desempenho possível, mas não alterando de nenhuma forma os sistemas operativos de raíz de qualquer das máquinas.

Isto porque se pretendia espelhar uma estrutura real, e não consolidar servidores ou apurar tecnicamente o desempenho de qualquer componente da arquitectura. Os sistemas operativos deviam assim ser usados obrigatoriamente inalterados.

Na verdade o problema principal que as soluções de virtualização têm para resolver é o tratamento das instruções privilegiadas sobre os recursos da máquina que aloja os sistemas virtuais, uma vez que essas instruções têm de ser interceptadas pelos chamados *Virtual Machine Monitors* (VMM) pois de outra forma são tratadas como instruções normais à vista do sistema operativo nativo e sem os privilégios necessários para aceder aos recursos, despoletando falhas *General Protection Faults* (GPF) [102]. Para contornar estes problemas nas arquitecturas IA32 surgiram variadas alternativas, umas interceptando tais instruções e tratando-as por chamadas a rotinas que se encarregam de gerir essas instruções, e outras alterando os sistemas operativos de forma a tirarem disso partido na extrapolação a níveis de desempenho melhorados, a chamada "Para-Virtualização", como é o caso do XEN da XenSource [70] cuja arquitectura se indica na Figura 4.4.

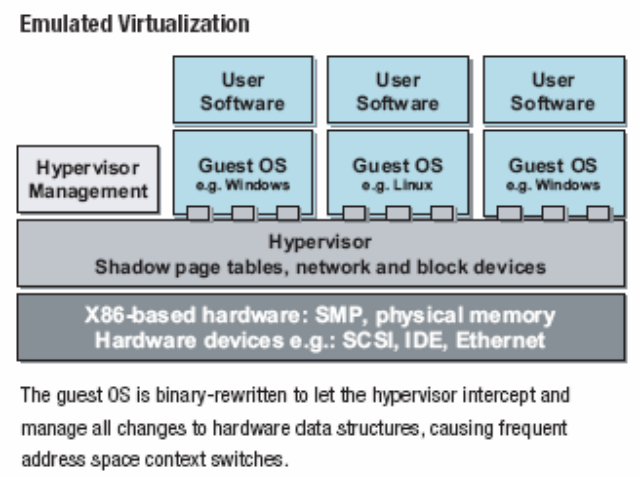


Figura 4.4 – Arquitectura XEN Source [111]

Relativamente às particularidades da virtualização da memória disponível, algumas das técnicas podem ir desde a recuperação de memória considerada menos importante pelo sistema operativo, à "taxação" de memoria de forma a distinguir qual a mais ou menos importante de forma a garantir a permanência do desempenho pretendido, entre outras abordagens [82].

Assim, sabendo logo à partida que esta selecção podia não contemplar ambientes com maiores indicadores de desempenho como é o caso do XEN, optou-se pelas soluções da VMware por não acrescentarem nenhum constrangimento ou limitação neste ponto, permitindo o uso dos sistemas operativos inalterados.

A escolha recaiu sobre a versão GSX Server cuja arquitectura se indica na Figura 4.5.

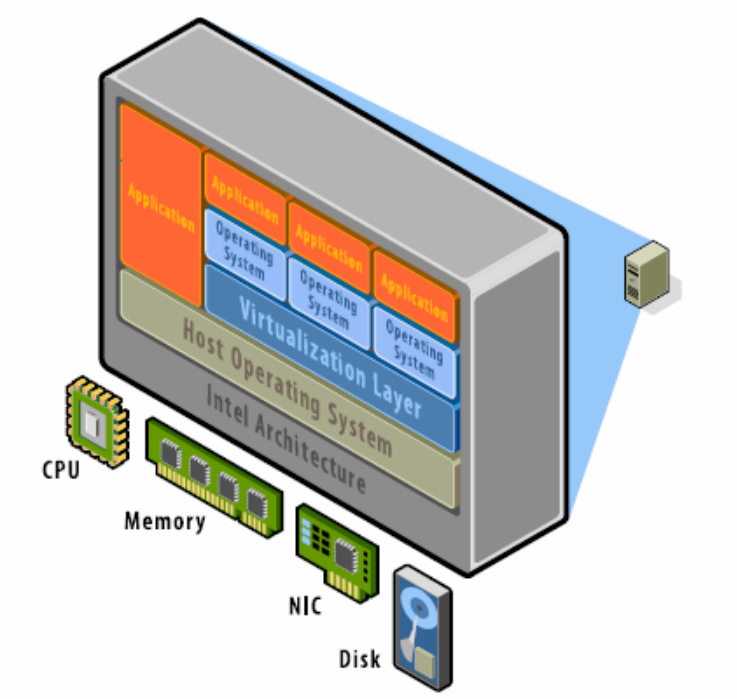


Figura 4.5 – Arquitectura VMware GSX Server 3 [107]

Entre as soluções da VMware, a baseada no ESX Server traria mais desempenho nativo por não necessitar da camada intermédia do sistema operativo, usando os seus próprios drivers.

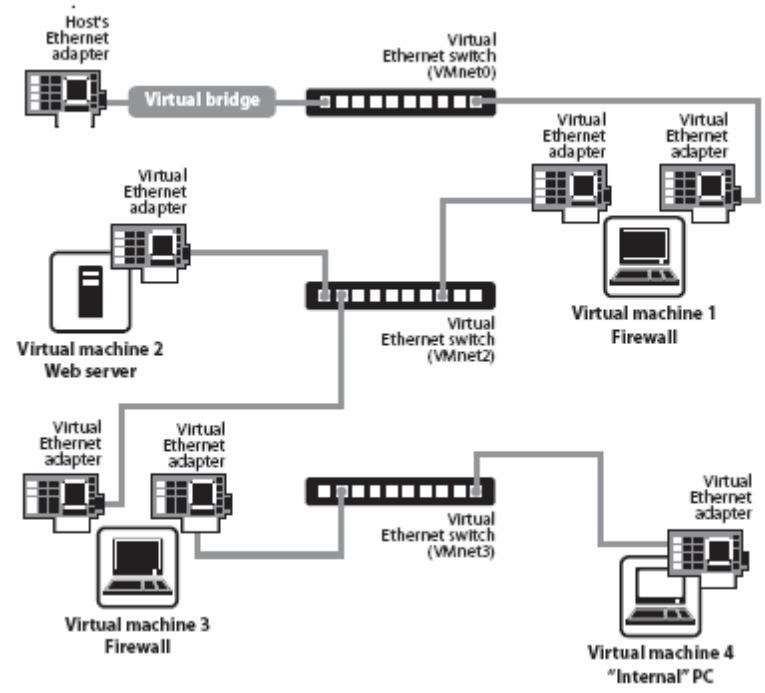


Figura 4.6 – Configurações de rede avançadas com VMware GSX. [104]

Na verdade a sua instalação é directamente executada sobre uma máquina "virgem" sem qualquer software e com os seus próprios drivers. No entanto apesar dos benefí-

cios que poderiam ser alcançados no desempenho, obrigaria a ter hardware dedicado apenas para o laboratório o que por razões logísticas não seria possível. Por outro lado, o maior ou menor desempenho, como já foi dito, não afectaria a análise do modelo em si. Outra das razões porque foi seleccionada esta ferramenta, prende-se com as capacidades de *networking* virtual que permite implementar. A Figura 4.6 ilustra esse facto, apresentando um exemplo retirado do manual de administração de uma possível configuração.

No caso do laboratório virtual aqui usado, foram extrapoladas ao máximo tais capacidades, tirando partido das várias interfaces de rede possíveis de implementar por máquina (até quatro por máquina), dos servidores DHCP integrados para cada comutador virtual, e das capacidades de *Network Address Translation* (NAT), *bridging* e *subneting* sobre os dez comutadores possíveis, usados para as redes do Centro de Dados, *Sites* remotos, Internet *gateway* e DMZ. A Figura 4.7 mostra o aspecto de algumas das configurações possíveis de realizar.

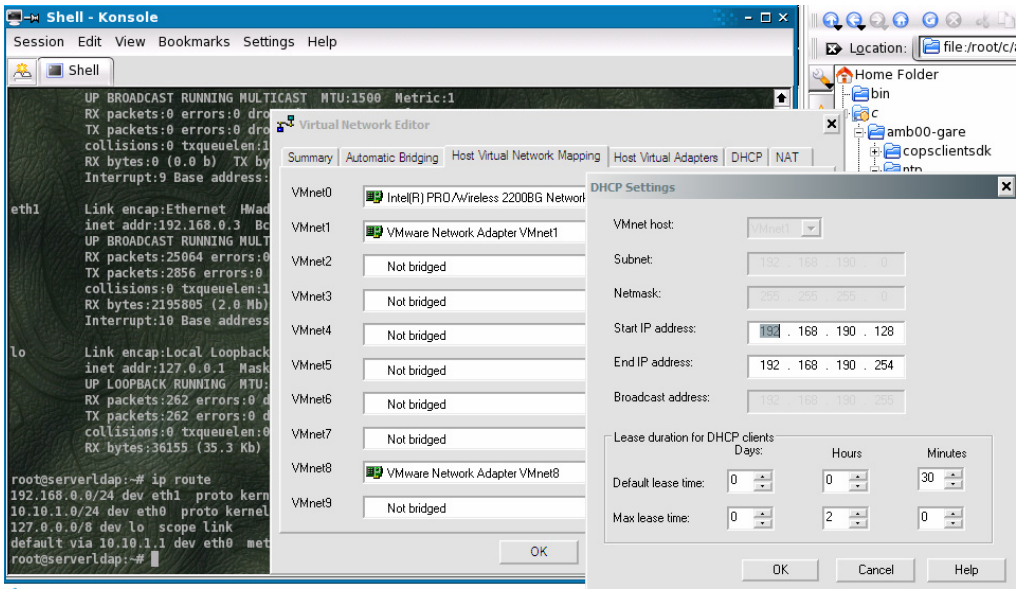


Figura 4.7 – Configuração de uma rede LAN e do servidor DHCP correspondente.

4.4.3 OpenLDAP

Entre as várias soluções disponíveis para o repositório, grande parte delas residem no domínio comercial com adaptações e desenvolvimentos próprios. Além disso, o facto de não ser possível conhecer de forma transparente as implementações, como o é em OpenSource, constitui uma desvantagem decisora.

A opção centrou-se na distribuição OpenLDAP [137] para Linux por necessitar de poucos recursos de processamento e memória, razão pela qual também foi posta de lado a opção do Solaris 10, e também por ser uma norma aceite pela comunidade nas suas várias vertentes de robustez e fiabilidade comprovada. Assim foi criado sob o servidor

serverldap.vnet.lab o servidor de directório do ambiente com o OpenLDAP 2.2.25 com os serviços necessários para a criação da organização detalhada no Capítulo 3, e para a gestão da infra-estrutura virtual.

Após terem sido criadas as entradas iniciais da organização já com uma estrutura mínima criada, usou-se o cliente de administração Softerra LDAP administrator por questões de produtividade [134]. A Figura 4.8 mostra o seu aspecto geral. Esta ferramenta foi de uma importância crucial na gestão do directório e em todos os trabalhos de teste e análise dos *schemas* e classes usadas.

Embora existissem algumas aplicações de gestão de um directório LDAP em Linux, nomeadamente a ferramenta GC entre outras, as suas funcionalidades estavam ainda longe das obtidas com o Softerra LDAP Administrator.

De notar que a ferramenta foi usada de "fora" do laboratório virtual acedendo a uma máquina do centro de dados virtual, razão pela qual a Figura 4.8 mostra um ambiente Windows e não Linux.

De qualquer forma a criação e gestão dos *schemas* implementados sobre o directório foram um trabalho manual a que esta ferramenta não deu apoio.

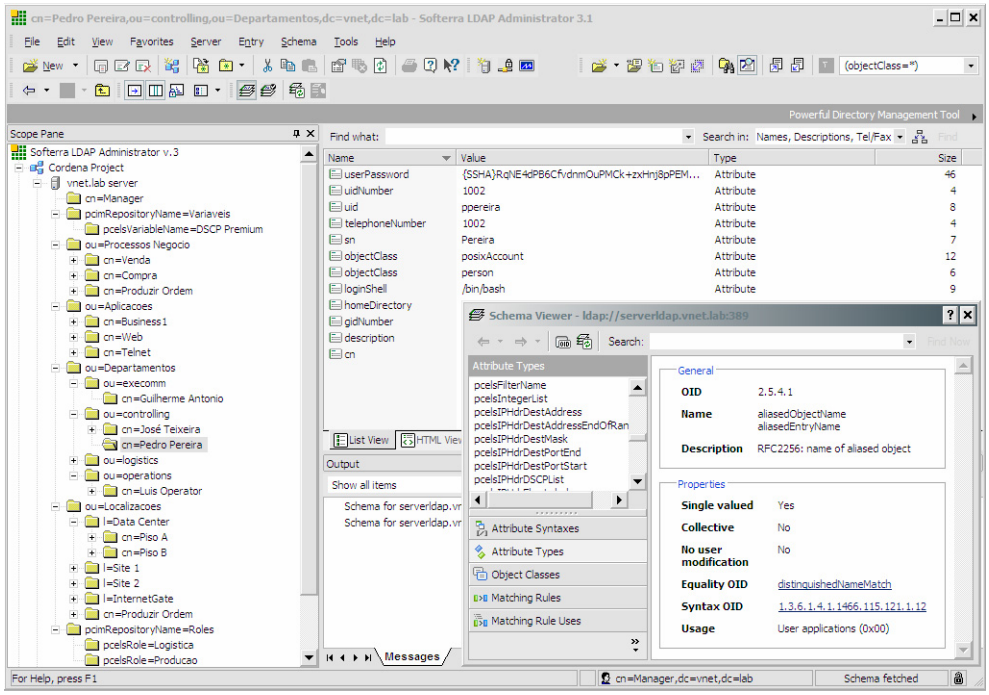


Figura 4.8 – Cliente de administração do directório LDAP em Windows.

4.4.4 Módulo iproute2

Uma das dificuldades deste trabalho foi a necessidade de encontrar uma forma óptima de representar *router*s sobre o laboratório virtual.

Foram testadas várias abordagens, nomeadamente as soluções de simulação da Cisco, o Quagga Software Routing Suite [100] baseado no Zebra *Router*, o *router* Freesco [132], entre outros.

No final optou-se por implementar os *router*s sobre a mesma distribuição Linux Slackware já indicada e pelas razões apontadas para a selecção da distribuição Linux e mais um importante: o facto de não se pretender implementar emuladores e simuladores, mas sim *router*s nativos.

Como foi impossível encontrar uma forma de o fazer no ambiente virtual, foram implementados *router*s Linux com base no módulo iproute2 [101] já disponível com as versões de *kernel* mais recentes (desde a versão 2.2) e que implementa todas as funções e mecanismos necessários à gestão de Qualidade de Serviço e de todas as entidades necessárias, marcadores, classificadores, mecanismos de *queuing*, etc...

De outra forma o objectivo inicialmente pretendido só conseguiria ser atingido se de alguma forma fosse possível aplicar uma imagem (um ficheiro *bin*) de um *router* no ambiente virtual, sendo necessário para isso arrancar essa imagem com o hardware de um computador que aloja o ambiente virtual, reconhecendo as interfaces de rede com as do *router* virtualizado. Isto ainda não foi possível!

Mais uma vez tais incursões serviram para entender claramente quais os blocos constituintes para os mecanismos de tradução, e não para lançar uma implementação da plataforma CORDENA. O facto de ter sido criado um ambiente útil de uma forma plural ao teste de variadas soluções mesmo sob outros domínios, faz com que o actual laboratório tenha já outras utilizações técnicas que não apenas as descritas neste trabalho.

4.4.5 Kits de desenvolvimento COPS-PR

A Intel disponibilizou um SDK para desenvolvimento de clientes COPS, COPS-PR em 2001 que hoje já não é suportado, ou pelo menos disponibilizado no seu *site* [127]. De qualquer modo o seu estudo foi crucial principalmente para o entendimento do funcionamento do protocolo conjuntamente com o RFC 2748 e 3084. A documentação de qualidade e detalhe incluída com o pacote foi de facto uma mais valia, comparativamente a outras distribuições com por exemplo a Vovida.com [131], na implementação da sua pilha COPS.

A implementação Intel apresentou um *kit* composto por camadas bem identificadas começando desde uma implementação de um módulo para portabilizar os desenvolvimentos entre Linux, Solaris e Windows, e seguindo pelas várias camadas protocolares tal como indicado na Figura 4.9, dando resposta a todos os tipos de mensagens definidas nas respectivas normas.

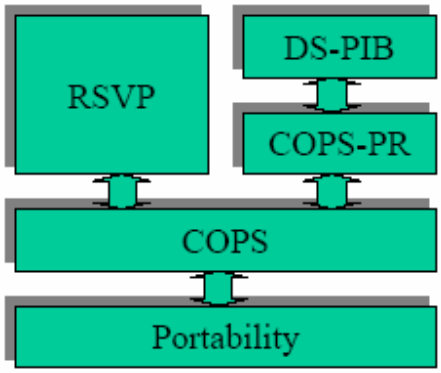


Figura 4.9 – Módulos do SKD COPS da Intel.[127]

Adicionalmente incluiu um gerador de PIB e um simulador funcional para teste e análise. A Figura 4.10, apresenta a sequência de mensagens cliente-servidor que tomam lugar entre o módulo COPS-PR e o módulo PIB – DiffServ.

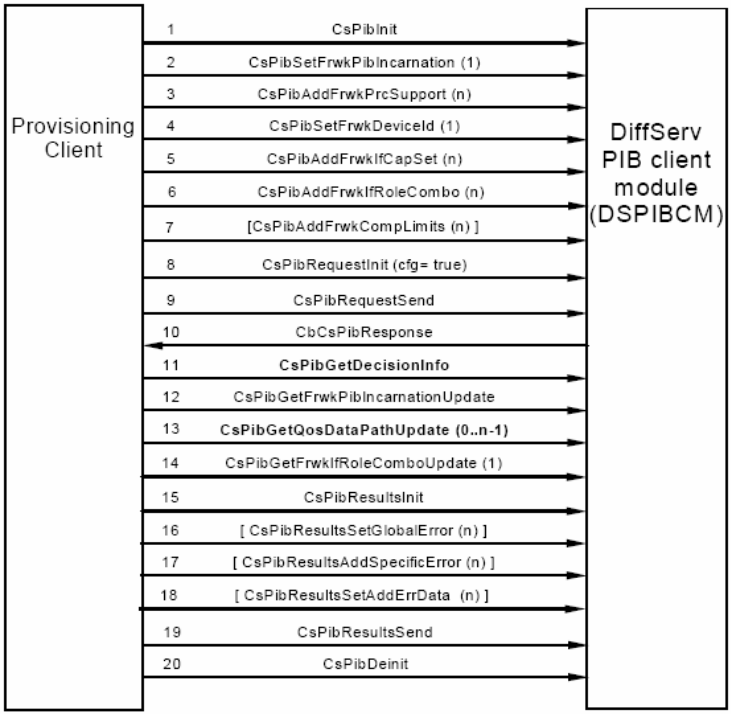


Figura 4.10 – Intel COPS-Comunicação entre Clientes Provisioning e Diffserv PIB [127]

4.4.6 Solaris 10

Embora exista no laboratório uma instância de Solaris 10, esta não se prendeu directamente aos trabalhos relacionados com a especificação da plataforma CORDENA, embora na fase inicial tenha sido equacionada para a implementação do servidor de directório. Não o foi por necessitar de mais recursos mínimos em termos de RAM e processamento, variável muito importante neste caso.

O seu uso prende-se com os testes relativos à tecnologia de compartimentação de processamento em zonas estanques, funcionalidade com a qual se pretende equacionar a aplicação e teste de políticas relativas à gestão das conexões às aplicações nos casos em que é necessário analisar soluções de tolerância a falhas, de transferência geral de utilizadores de uma para a outra instância, etc.

Nas Zonas existe apenas uma instância do *kernel*, mas o sistema operativo é dividido entre várias zonas isoladas entre si criando ambientes de execução perfeitamente estanques que podem ser arrancados, reiniciados, parados independentemente uns dos outros [98] [102]. A Figura 4.11 ilustra esta abordagem.

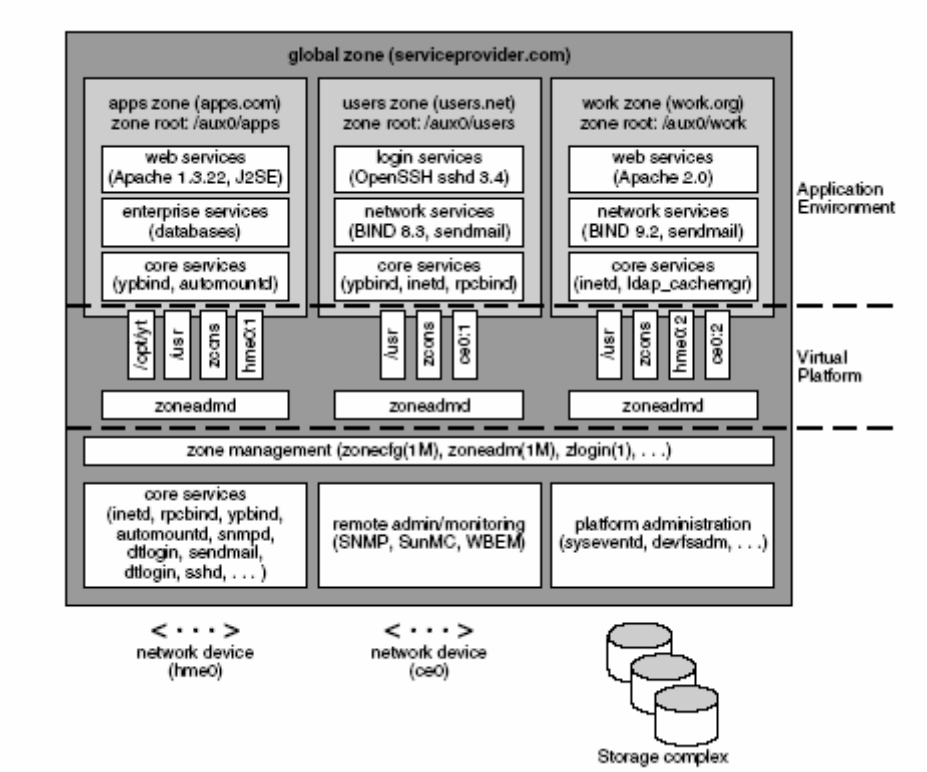


Figura 4.11 – Consolidação de zonas em Solaris 10 [98].

A funcionalidade de virtualização incluída no Solaris 10 através dos chamados *Trusted Containers* ou *Zones* e o seu uso na aplicação de políticas, foi a razão da sua inclusão nos servidores serverapp1.vnet.lab e serverapp2.vnet.lab.

4.4.7 Tcl/tk

Tcl-Tk é uma linguagem de *scripting* muito útil para a prototipagem de aplicações com um tempo de aprendizagem curto e que permite desenvolver rapidamente protótipos aplicacionais. Conta ainda com implementações das suas bibliotecas multiplataforma permitindo a passagem dos desenvolvimentos de um para o outro sistema operativo.

Neste caso foi usada para criar uma interface simples para correr os *scripts* principais sob o ambiente, nomeadamente arranque e paragem de processos, listagens, consulta de estado, etc., contribuindo por isso para uma maior produtividade ao invés de o trabalho ser repetidamente feito por linha de comando. A Figura 4.12 mostra uma janela da aplicação.

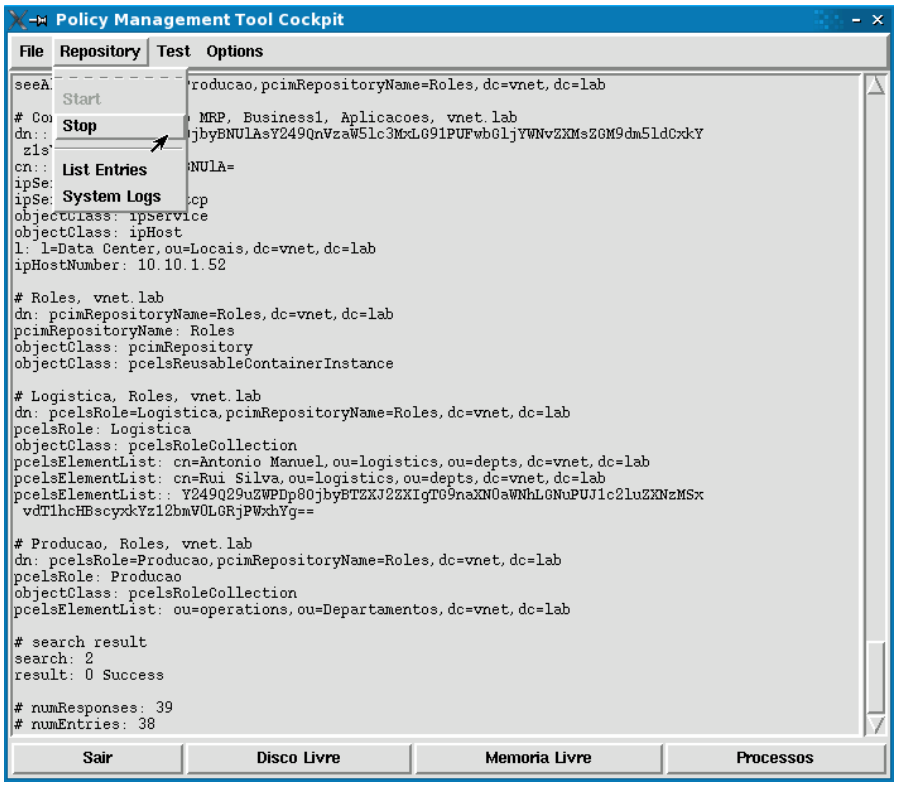


Figura 4.12 – Aplicação de manutenção do ambiente em Tcl-tk.

Existem disponíveis centenas de bibliotecas disponíveis na Internet [135] [136] que permitem interagir com todas as partes constituintes das infra-estruturas, desde bibliotecas com as principais funções necessárias para interagir com um servidor LDAP, com a rede, com o sistema operativo, etc.

Tcl/Tk foi também a cola que uniu alguns *scripts* desenvolvidos em *Shell Scripting* [113] [114].

4.4.8 Bind 9.2

O uso de um domínio interno de resolução de nomes foi importante de forma a criar um ambiente composto por todas as partes constituintes de uma infra-estrutura de rede e garantir as funcionalidades necessárias ao ambiente. Nesse sentido seleccionou-se o pacote Bind 9.2 [99] para implementar o processo de resolução de nomes, por ter por

um lado fiabilidade comprovada e por ser a norma aceite também por toda a comunidade. O domínio de nomes criado foi o vnet.lab.

A Figura 4.13 apresenta uma versão das entradas criadas no laboratório para as máquinas mais importantes. As configurações totais estão patentes no Anexo respectivo.

```
$TTL 86400
@       IN  SOA serverldap.vnet.lab. root.vnet.lab. (
                100 ;Serial
                3H ; refresh
                15M ; retry
                1W ; expiry
                1D ) ; minimum

        IN  NS  serverldap.vnet.lab.
        IN  MX 10   vnet.lab.

localhost   IN  A   127.0.0.1
vnet.lab    IN  A   10.10.1.50
routerdc    IN  A   10.10.1.1
PEP1        IN  A   10.10.1.1
serverldap  IN  A   10.10.1.50
PR          IN  A   10.10.1.50
servermain  IN  A   10.10.1.51
PMT         IN  A   10.10.1.51
dns         IN  A   10.10.1.50
serverapp1  IN  A   10.10.1.52
serverapp2  IN  A   10.10.1.53
dns2        IN  A   10.10.1.54
solaris10   IN  A   10.10.1.54
mail        IN  A   10.10.1.54
serverapp3  IN  A   10.10.1.54
routernet   IN  A   10.10.4.1
routerst1   IN  A   10.10.2.1
routerst2   IN  A   10.10.3.1
PEP2        IN  A   10.10.2.1
PEP3        IN  A   10.10.3.1
PEP4        IN  A   10.10.4.1
```

Figura 4.13 – Registos do servidor BIND 9 - dns.vnet.lab

4.4.9 Notas Finais

Muito mais haveria que falar descendo alguns níveis nas configurações e nas experiências guardadas nos testes efectuados neste laboratório. Ficam aqui aquelas que de alguma forma constituem um acréscimo de conhecimento significativo e útil para outros trabalhos. Houve também uma interessante convivência entre tecnologias já com algum tempo e outras relativamente recentes, o que prova a complementaridade das soluções sejam elas o uso de uma linguagem como Tcl/tk para prototipagem, ou técnicas de virtualização de ambientes.

4.5 Proposta de Planeamento

Todas as fases de desenvolvimento da uma plataforma devem recorrer às ferramentas de gestão habituais de projectos, por facilitarem e mensurarem a gestão das acções, nomeadamente nos prazos, tarefas, recursos, implicações concretas, etc.

Apresenta-se de seguida um modelo de planeamento apenas como uma referência futura de implementação. Neste caso o trabalho necessário foi dividido em quatro fases principais que se passam a descrever:

- Planeamento
- Especificação
- Investigação e pesquisa
- Implementação

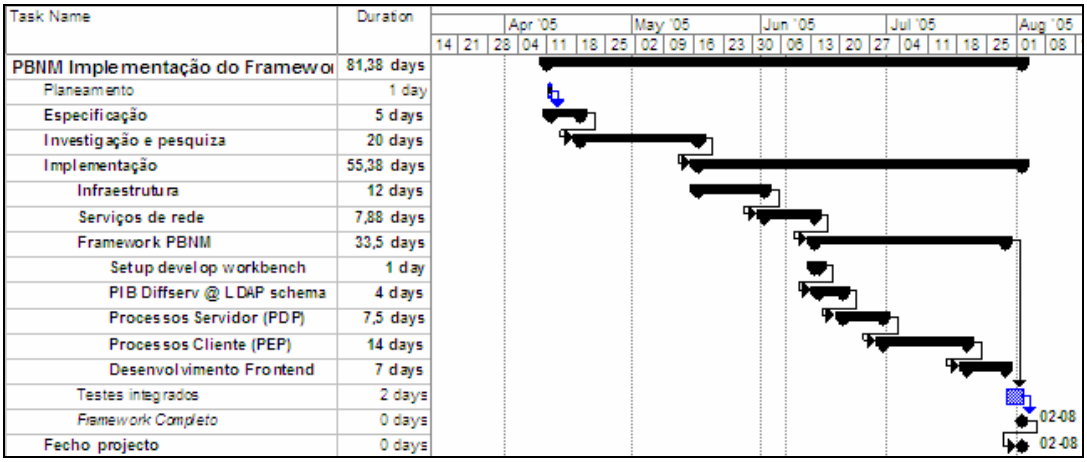


Figura 4.14 – Gráfico de Gantt do Plano de Projecto

Salienta-se que os tempos indicados no projecto, são uma aproximação uma vez que dependem muito da equipa de projecto e da sua perícia nos domínios tecnológicos em

questão. De qualquer forma, o plano apresentado na Figura 4.14 poderá ser usado em projectos de âmbito idêntico. Salienta-se que algumas das tarefas indicadas poderiam recorrer a relações diferentes, nomeadamente *Start-to-Start* ("começo simultâneo de duas tarefas") caso existissem recursos ou equipas a trabalhar em simultâneo, escalonamento possível na fase de desenvolvimento da plataforma por exemplo.

No total a previsão apontou para um duração da implementação de 81 dias consumidos entre 55 para a implementação propriamente dita, 20 em investigação, 5 em especificação, e o restante em planeamento.

Assumindo que os recursos disponíveis apresentam um domínio alargado sob as tecnologias envolvidas.

4.5.1 Planeamento

A primeira fase é relativamente curta, mas de uma importância inversamente proporcional à sua duração.

De qualquer forma, sugere-se esta abordagem para um projecto mesmo de uma equipa pequena, por sistematizar e relacionar as tarefas de desenvolvimento, e de captar ideias e necessidades que de outra forma não chegam a emergir do decurso do trabalho propriamente dito.

4.5.2 Especificação

Entende-se que uma mais valia de qualquer projecto de desenvolvimento, é a necessidade de bem especificar. Uma das formas de o conseguir ou pelos menos de não o prejudicar, é estancar a especificação funcional e a descrição dos objectivos pretendidos, da miríade de variáveis e configurações técnicas que também será necessário analisar mas em sede própria.

Um dos erros mais habituais é limitar ou acrescentar determinados constrangimentos tomando por base limitações previstas *à priori* sobre áreas técnicas ou de indisponibilidade de recursos. É imperativo saber onde queremos ir, e qual é no nosso entender o objectivo mais nobre possível dentro do âmbito que estabelecemos.

Limitações e constrangimentos são outros problemas que nada têm a haver com especificação e objectivos. O seu tratamento é imperativo, mas no seu lugar como distintos pontos a trabalhar, gerir, assumir, etc, tal como muitos outros que irão surgir ao longo do desenrolar do projecto.

É dentro deste pressuposto que a fase de especificação deve emergir com a abrangência necessária e suficiente sobre os domínios de especificação funcional e técnica. A fase de especificação debruça-se assim sobre todas as definições necessárias à plata-

forma sem entrar em linha de conta com as particularidades técnicas das soluções encontradas. Foram previstos 5 dias para completar a especificação técnica e funcional.

É um trabalho bastante orientado aos objectivos, sem contemplar as eventuais dificuldades na implementação de tais requisitos. São na verdade dois domínios que devem ser bem separados desde o início.

4.5.3 Investigação e pesquisa

Esta fase, sem dúvida a segunda maior em duração serve para investigar quais as melhores alternativas para a implementação da plataforma. É neste período que se envereda por implementar todo o ambiente de testes sob uma infra-estrutura virtual (no caso CORDENA), pois foi a forma de conseguir criar um ambiente real alargado para testar o plataforma de gestão numa infra-estrutura representativa das existentes na maior parte das empresas.

De uma forma geral é feito todo a procura de alternativas e soluções para todas as partes constituintes do trabalho.

4.5.4 Implementação

Há como de resto seria de esperar um maior detalhe na lista das tarefas do projecto nestas fases. É muito importante que o plano seja a linha orientadora dos trabalhos a realizar e da sua sequência. É com base nessa importância que o plano deve ser elaborado.

São contempladas fases de teste intercalares para solidificar os problemas encontrados em cada parte de desenvolvimento, e no final uma fase final de testes integrados.

A fase de implementação divide-se pelas seguintes etapas:

- Instalação do "*Develop Workbench*" - ou ambiente de trabalho para desenvolvimento.
- A implementação do modelo de dados usado para registar e tratar políticas (LDAP, PIB, respectivos *schemas*, etc)
- Desenvolvimento dos processos do servidor
- Desenvolvimentos dos processos do cliente
- Interacção cliente - servidor
- Desenvolvimento do *frontend*
- Testes integrados
- Finalização do projecto.

Normalmente, deverá ser enquadrada uma última fase de fecho de projecto que inclua as tarefas de fecho, que podem ir desde ajustes contratuais, divulgação, arquivo de desenvolvimentos, fases de teste alargado de versões alfa e beta, etc.

4.5.5 Notas Finais

Destaca-se apenas uma breve nota relativa e este exercício de planeamento para descrever o carácter aproximado do modelo.

A metodologia depende bastante da equipa de projecto definida para uma implementação deste tipo, assim como todos as definições típicas de um qualquer projecto, além da equipa, o orçamento, as datas previstas para entrega dos produtos e resultados. Pretende-se sim, abordar uma filosofia de trabalho por projecto, entendida como crucial para obter resultados e chegar a bom porto. Filosofia que acaba por capacitar o trabalho em equipa e a satisfação por cortar a meta em conjunto.

5 Conclusões

Embora muitas das constatações importantes tenham sido redigidas ao longo do trabalho, quer nas notas finais dos Capítulos ou Secções, quer nas perspectivas de aplicação das tecnologias descritas, far-se-à adicionalmente uma análise de algumas conclusões adicionais nas suas diversas componentes.

De forma a facilitar a sua leitura, tais conclusões foram divididas pela tecnologia em si (PBNM), do modelo proposto, dos trabalhos práticos, finalizando com uma perspectiva sobre oportunidades de trabalho futuro.

5.1 Acerca da Gestão de Redes Baseada em Políticas

PBNM é na realidade uma filosofia de gestão de redes diferente e com vantagens comprovadas.

O facto de definir as políticas a um nível de abstracção elevado cria vantagens, quer no nível técnico ao simplificar o tratamento e administração das estruturas, quer ao nível funcional e empresarial, ao aproximar a utilização da estrutura dos objectivos de negócio e das políticas da empresa.

Ideologicamente os objectivos desta tecnologia são nobres e ambiciosos. Tecnicamente ainda há muito caminho a percorrer até estabilizar o modelo, quer na normalização necessária a todos os níveis, quer na adopção pelos fabricantes de suporte PBNM nativo, aos equipamentos que comercializam.

PBNM pode também ser uma oportunidade para conciliar os esforços do IETF e o DMFT, em termos de trabalho conjunto e da definição de normas que satisfaçam a maioria das comunidades interessadas.

As primeiras necessidades para a utilização de PBNM surgiram com os requisitos para controlar QoS, com implementações IntServ via reservas com o protocolo RSVP, e como vimos no último Capítulo 2, o suporte de serviços e aplicações com QoS é uma área privilegiada para aplicar PBNM. No entanto, também como retratado, as vantagens de integração, simplificação e homogeneização de configurações por todo um domínio de controlo, não são desprezáveis.

Veja-se como exemplo a relação entre endereços IP e utilizadores, fazendo cair a distância entre as ACL e os directórios de utilizadores ao nível do sistema operativo de rede. Esta integração de objectos trará certamente valor à tecnologia.

Finalmente, é com a integração de outras fontes de informação, nomeadamente os directórios existentes nas organizações, que as potencialidades desta tecnologia podem ser extrapoladas.

A oportunidade é a criação de uma plataforma que permita estender as capacidades nativas de cada parte constituinte para lá das suas atribuíções iniciais, partindo para uma Gestão Baseada em Políticas totalmente matricial, mapeando os sistemas e tecnologias de informação aos processos de negócio dentro das organizações.

5.2 Acerca do Modelo CORDENA

É útil antes de mais um regresso ao final do Capítulo 3, onde são tecidas em algum detalhe as perspectivas de aplicação, quer de uma forma geral, quer no que respeita ao apoio a alguns problemas bem conhecidos e actuais relativamente ao controlo de SLA/SLS, redundância e tolerância a falhas, etc. Por essa razão esta secção não vai certamente repetir o que aí oportunamente foi levantado. Pretende-se sim dar um enquadramento adicional desta tecnologia nos dias de hoje.

Repare-se antes de mais que as temáticas relacionadas com a Gestão Baseada em Políticas tiveram início nos aos 80 [69], mas sem conseguir afirmar. O propósito foi já na altura interessante, mas eventualmente a tecnologia não estava ainda com a maturidade suficiente para avançar com um modelo completo e funcional que a implementasse no seu todo.

Passou-se o mesmo, como já foi referido, com as técnicas de virtualização iniciadas nos anos 70. Na altura apenas como solução para evitar que o desenvolvimento aplicacional estivesse dependente dos problemas do hardware. Um enquadramento bem diferente dos dias de hoje.

Chegamos assim ao Séc. XXI com a tecnologia PBNM já com a maturidade suficiente, e na verdade aqui e ali vamos vendo alguns produtos com características de PBNM nas ofertas dos principais fornecedores de soluções em termos globais.

O surgimento dos sistemas de directório, dos protocolos de transporte de políticas, a evolução dos equipamentos, da sua capacidade de processamento e da miniaturização dos circuitos, deve ter sido um factor que contribuíu para esta maturidade referida.

É por estas razões que se defende que a plataforma CORDENA e as suas fundações nesta tecnologia se justificam cada vez mais. Assim, por que não tirar partido das vantagens enumeradas nas secções relacionadas com as perspectivas de aplicação?

Na verdade, como já foi referido oportunamente, estas migrações tecnológicas não ocorrem de um dia para o outro, pois se por um lado as incursões científicas provam a sua aplicabilidade e potencial, o mercado terá sempre um ciclo de cerca de quatro anos na adopção das tecnologias de uma forma generalizada. Referem-se quatro anos apenas pelos ciclos de amortização e depreciação financeira de equipamentos, pois em grande parte das situações, os momentos de renovação tecnológica associadas às aplicações financeiras, são os momentos oportunos para equacionar novas tecnologias, de preferência com a maturidade necessária para suportar os processos de negócio críticos sem turbulência adicional.

Por outro lado, a homogeneização do mercado leva tempo na adopção destas novas tecnologias. Nestes casos a adaptabilidade do mercado vai ser posta à prova. Irá haver necessidade de menos trabalho especializado nesta ou naquela tecnologia, para passar

a haver no desenvolvimento das plataformas de gestão, de teste e desenvolvimento de políticas, ou de gestão dos processos críticos e não apenas para as tarefas operacionais.

Não nos alargaremos mais nestas abordagens, pois rapidamente nos afastamos do âmbito do trabalho. De referir apenas, cumulativamente ao que já foi dito, que se entende estarem as condições criadas para avançar com estas tecnologias, das quais a solução CORDENA é uma modelação arquitectural com as peças constituintes necessárias com vista a apoiar uma implementação funcional de uma Plataforma de Gestão de Redes Baseada em Políticas.

5.3 Acerca das Incursões Experimentais

Foram apresentadas, ao longo do Capítulo 4, as tecnologias mais importantes usadas no decurso dos trabalhos efectuados da plataforma CORDENA, com especial destaque para o laboratório virtual usado.

Entende-se que estas experimentações têm o seu valor acrescentado pela facilidade com que se desenhou, testou e implementou uma infra-estrutura de teste em pouco tempo e sem constrangimentos de maior. Esta é uma das conclusões que podemos tirar desde já, a facilidade e flexibilidade destes ambientes.

Apesar das tecnologias de virtualização já estarem no meio e no mercado desde os anos 60 [102], só nos dias de hoje ganharam uma aplicabilidade e divulgação maior graças naturalmente à evolução tecnológica em toda a linha e aos progressos alcançados. A secção relativa à virtualização refere a este nível as decisões e escolhas efectuadas, abordando também outras técnicas de virtualização que noutros contextos poderão ser mais aprofundadas. Dependendo das necessidades de aplicação, diferentes ferramentas poderão ser a escolha acertada.

Para lá das partes mais relacionadas com a infra-estrutura global, de destacar a facilidade de utilização e integração da distribuição OpenLDAP com o restante ambiente. Após a resolução de alguns problemas técnicos naturais na instalação, a ferramenta funcionou muito bem sem qualquer falha ou problema. Claro está que a dimensão é um factor importante por neste caso não irmos além de algumas dezenas de entradas sobre o directório.

Finalmente é apresentada uma pequena proposta de planeamento da implementação num âmbito mais restrito. Um dos pontos importantes é a metodologia e não apenas os tempos decorridos ou os recursos envolvidos. O plano apresentado é apenas um esboço metodológico para a implementação da plataforma.

Na verdade, embora não estejamos a falar de uma implementação prática ou da prototipagem da plataforma no seu todo, constatamos que os ambientes criados cumpriram em primeiro lugar a sua função por permitir durante todo o trabalho, testar, implementar, apagar, recriar, etc, várias componentes do ambiente. Por outro lado, as experiências colhidas e as decisões mais importantes relativamente às principais tecnologias são também registadas por serem úteis para o leitor.

Uma última palavra relativamente ao desenvolvimento e teste de políticas. De facto, constatou-se que o teste e desenvolvimento de políticas, pode passar pelo uso de sistemas virtualizados, sendo desde já uma constatação deste trabalho. Desta forma, é possível testar o uso das plataformas de gestão de políticas sobre ambientes gémeos dos sistemas reais, sem qualquer limitação quer das próprias instâncias quer das funcionalidades em si.

5.4 Resumo das Contribuições Científicas

Após ter enquadrado o leitor com a tecnologia em causa, a actual dissertação apresentou a Modelação de uma Plataforma de Gestão de Redes Baseada em Políticas - A Plataforma CORDENA.

Enumeram-se de seguida as principais contribuições descritas, maioritariamente relacionadas com as especificações e definições da plataforma:

- Modelação de uma Organização de Base que serviu de génese aos objectos organizacionais que povoaram o directório com informação.
- Definição de um Modelo de Dados enriquecido com abordagens próprias para auxiliar a tradução de entidades da organização para variáveis de rede.
- Definição do ambiente operacional da plataforma, e do seu fluxo de processos e operações de alto nível.
- Modelação da arquitectura subjacente desde os repositórios de informação (objectos organizacionais e políticas), até aos mecanismos mergulhados na rede.
- Descrição dos mecanismos de tradução de utilizadores, grupos e departamentos, localizações físicas, ambientes aplicacionais, processos de negócio, variáveis temporais e variáveis inerentes aos equipamentos terminais para variáveis de rede, nomeadamente a necessária para alimentar os protocolos de

transporte de políticas e os PIB´s existentes para determinadas tecnologias, no caso para o exemplo dos Serviços Diferenciados.

- Descrição de um conjunto de considerações importantes relacionadas com a adopção da tecnologia em si, e especificamente no que respeita à definição e controlo de SLA/SLS, na tolerância a falhas, na resiliência da rede e na gestão de conflitos na aplicação de políticas.
- Descrição de um Modelo Humanizado da Rede como forma de melhorar o desempenho e a escalabilidade do modelo nos patamares de rede.
- Registo das principais decisões e conclusões relacionadas com o uso de várias tecnologias experimentais no estudo, desenvolvimento e teste das abordagens aqui descritas, nomeadamente o sistema operativo Linux, a virtualização de hardware, a distribuição OpenLDAP, entre outros.
- Descrição de um laboratório completo virtualizado com o qual se sugere o teste e desenvolvimento de políticas.
- Descrição de uma proposta de planeamento para uma eventual implementação da Plataforma CORDENA.

Para lá de outras considerações tecidas oportunamente nos diversos Capítulos, estas foram as principais contribuições deste trabalho.

5.5 Trabalho futuro

Poderá dizer-se que este trabalho é de certa forma rico em oportunidades de desenvolvimento laterais para lá da implementação central da Plataforma CORDENA.

Assim, uma vez que este trabalho é uma abordagem de especificação e modelação, o uso dela numa implementação aplicacional integrada, constitui a maior oportunidade mas também a mais extensa que merece ser bem enquadrada sob qualquer cenário.

De qualquer forma, é útil descer um nível na arquitectura, o que nos abre várias oportunidades indexadas aos diversos módulos constituintes, de forma que algumas de entre muitas possibilidades serão por exemplo:

- Desenvolvimento de módulos para construção e encapsulamento das mensagens em COPS-PR, NSIS, CORBA, etc.

- Desenvolvimento de Módulos para outras aplicações que não apenas por QoS, como por exemplo para Segurança Integrada, etc.
- Desenvolvimento de um módulo para Gestão de Conflitos usando técnicas de extracção de conhecimento em rede e inteligência artificial.

Muitas mais poderiam ser levantadas, pelo menos no que respeita às várias partes constituintes da arquitectura, por exemplo o Ciclo de Controlo, os Monitores, etc, etc...

Fica, no entanto, como fulcral a oportunidade da implementação da Plataforma como trabalho futuro, dependendo das oportunidades e a sua prossecução com mais ou menos detalhe e com mais ou menos abrangência.

6 Índice de Referências

6.1 RFC e outros Standards

6.1.1 IETF - Policy Framework

[1] RFC 3198 - Terminology for Policy- Based Management

[2] RFC 3060 - Policy Core Information Model -- Version 1 Specification

[3] RFC 3460 - Policy Core Information Model (PCIM) Extensions

[4] RFC 3644 - Policy Quality of Service (QoS) Information Model

[5] RFC 3670 - Information Model for Describing Network Device QoS Datapath Mechanisms

[6] RFC 3703 - Policy Core Lightweight Directory Access Protocol (LDAP) *Schema*

[7] RFC 4104 - Policy Core Extension LDAP *Schema* (PCELS)

[8] Policy QoS LDAP *Schema* (draft-ietf-policy-qos-*schema* -01)

6.1.2 IETF - Resource Allocation Protocol

[9] RFC 2748 - The COPS (Common Open Policy Service) Protocol

[10] RFC 3084 - COPS Usage for Policy Provisioning (COPS-PR)

[11] RFC 3159 - Structure of Policy Provisioning Information (SPPI)

[12] RFC 3318 - Framework Policy Information Base

[13] RFC 3483 - Framework for Policy Usage Feedback for COPS-PR

[14] RFC 3571 - Framework Policy Information Base for Usage Feedback

[15] RFC 3520 - Session Authorization Policy Element

[16] RFC 3317 - Differentiated Services Quality of Service Policy Information Base

6.1.3 IETF - Network (LDAP)

[17] RFC 1274 - The COSINE and Internet X.500 *Schema*

[18] RFC 2079 - Definition of an X.500 Attribute Type and an Object Class to Hold Uniform Resource Identifiers (URIs)"

[19] RFC 2247 - Using Domains in LDAP/X.500 Distinguished Names

[20] RFC 2251 - Lightweight Directory Access Protocol (v3)

[21] RFC 2252 - LDAP *Schema* - Attibute Syntax Definitions

[22] RFC 2253 - LDAP (v3): UTF-8 String Representation of Distinguished Name

[23] RFC 2254 - The String Representation of LDAP Search Filters

[24] RFC 2255 - The LDAP URL Format

[25] RFC 2256 - A Summary of the X.500(96) User *Schema* for use with LDAPv3

[26] RFC 2293 - Representing Tables and Subtrees in the X.500 Directory

[27] RFC 2294 - Representing the O/R Address Hierarchy in the X.500 Directory Information Tree

[28] RFC 2307 - An Approach for Using LDAP as a Network Information Service

[29] RFC 2377 - Naming Plan for Internet Directory-Enabled Applications

[30] RFC 2589 - Lightweight Directory Access Protocol (v3): Extensions for Dynamic Directory Services

[31] RFC 2596 - Use of Language Codes in LDAP

[32] RFC 2649 - An LDAP Control and *Schema* for Holding Operation Signatures

[33] RFC 2696 - LDAP Control Extension for Simple Paged Results Manipulation

[34] RFC 2713 - *Schema* for Representing Java™ Objects in an LDAP Directory

[35] RFC 2714 - *Schema* for Representing CORBA Object References in an LDAP Directory

[36] RFC 2798 - Definition of the inetOrgPerson LDAP Object Class

[37] RFC 2829 - Authentication Methods for LDAP

[38] RFC 2830 - Lightweight Directory Access Protocol (v3): Extension for Transport Layer Security

[39] RFC 2849 - The LDAP Data Interchange Format (LDIF) - Technical Specification

[40] RFC 2891 - LDAP Control Extension for Server Side Sorting of Search Results

[41] RFC 3045 - Storing Vendor Information in the LDAP root DSE

[42] RFC 3062 - LDAP Password Modify Extended Operation

[43] RFC 3088 - OpenLDAP Root Service: An experimental LDAP referral service

[44] RFC 3112 - LDAP Authentication Password *Schema*

[45] RFC 3296 - Named Subordinate References in LDAP

[46] RFC 3377 - Lightweight Directory Access Protocol (v3): Technical Specification

[47] RFC 3383 - IANA Considerations for the LDAP

[48] Smith, M. et al, IETF Draft - The C LDAP Application Program Interface, 17 Nov 2000

6.1.4 IETF - Network (QoS - IntServ & DiffServ)

[49] RFC 1349 - Type of Service in the Internet Protocol Suite

[50] RFC 1633 - Integrated Services in the Internet Architecture

[51] RFC 2205 - RSVP Functional Specification

[52] RFC 2209 - RSVP Message Processing Rules

[53] RFC 2210 - The Use of RSVP with IETF Integrated Services

[54] RFC 2211 - Specification of the Controlled-Load Network Element Service

[55] RFC 2212 - Specification of Guaranteed Quality of Service

[56] RFC 2206 - RSVP Management Information Base using SMIv2

[57] RFC 2207 - RSVP Extensions for IPSEC Data Flows

[58] RFC 2208 - RSVP Applicability Statement Some Guidelines on Deployment

[59] RFC 2474 - Definition of the DS Field in the IPv4 and IPv6 Headers

6.1.5 DMTF

[60] DSP108 - CIM Policy Model White Paper v2.7, 18-June-2003

[61] DSP0123 - CIM Core Model V2.5 LDAP Mapping Specification

[62] DSP0124 - DMTF LDAP *Schema* for the CIM v2.5 Physical Information Model

[63] Bumpus, Winston, DMTF President, CIM Directory Mappings, DMTF 2002 Developers' Conference, June 10-13 2002

6.2 Bibliografia

[64] Carter, G., LDAP System Administration, O'Reilly, Mar 2003

[65] Bovet, D., Cesati, M., Understanding the Linux *Kernel*, O'Reilly, Oct 2000

[66] Ward, Brian, The book of VMware – The complete guide to VMware workstation, William Pollock, 2002

[67] Cisco Systems Inc, Cisco CCIE Fundamentals Network Design

[68] Kosiur, D. et al., Understanding Policy-Based Networking, Wiley

[69] Rocha, F., PBNM volta a dar cartas, Revista REDES nº 84, pág. 58-61, Abril 2002

[70] Barhamj, P., et al, Xen and the Art of Virtualization, University of Cambridge, ACM SOSP03

[71] Ponnappan, Appan et al, A policy based QoS management system for the intserv-diffserv based Internet, Siemens, IEEE POLICY'02 Proceedings

[72] Badr, N. et al, Policy-Based Autonomic Control Service, Liverpool John Moores University, IEEE POLICY'04 Proceedings

[73] Cuervo, Fernando, Policy Control Model a Key Factor for the Success of Policy in Telecom applications, Alcatel, IEEE POLICY'04 Proceedings

[74] Carvalho, P., Lima, S., Parada, C., Fontes, F., Carapinha, J., Comparação de Plataformas para Suporte de Serviços Diferenciados em Redes IP, Universidade do Minho, Portugal Telecom Inovação

[75] Trimintzios, P. et al, Policy-Based Network Dimensioning for IP Differentiated Services Networks, University of Surrey, University of Thessaloniki, University College London

[76] NESTOR An Architecture for Network Self-Management and Organization.pdf

[77] AUTONOMIA An Autonomic Computing Environment.pdf

[78] Towards Policy-Based Management of Active Networks.pdf

[79] Reyes, Maria A., Contributions to an advanced design of a Policy Management System, Verão 2003

[80] Polyrakis, A., Boutaba, R, The Meta-Policy Information Base, IEEE Network, March/April 2002

[81] Carvalho, Paulo; Santos, Alexandre; Solange, Lima; Vasco, Freitas, "Distributed Admission Control Model for CoS Network using QoS & SLS Monitor", 2003

[82] Waldspurger, C., Memory Resource Management in VMware ESX Server, Vmware Inc., Proceedings of the 5th Symposium on Operating Systems Design and Implementation, Boston, USA, Dec 2002

[83] Javvin Technologies Inc, Network Protocols Handbook, 2004

[84] Kuznetsov, Alexey, IP Command Reference, Institute of Nucler research, Moscow, 1999

[85] Sheridan-Smith, N, A Distributed Policy-based Netowrk Management (PBNM) system for Enriched Experience Networks (EENs), Doctoral Assessment, University of Technology, Sydney, Nov 2003

[86] Schonwalder, J. et al, On the future of the Internet Management Technologies, IEEE Communications Magazine, Oct 2003

[87] Flegkas, Paris et al, A Policy-Based Quality of Service Management System for IP Diffserv Networks, IEEE Network, March/April 2002

[88] Tsarouchis, C et al, A Policy-Based Management Architecture for Active and Programmable Networks, IEEE Network, May/June 2003

[89] Eddie Law et al, Scalable Design of a Policy- Base Management System and its Performance, University of Toronto, IEEE Communications Magazine, June 2003

[90] Nguyen, Thi Mai Trang; Boukhatem, Nadia, "COPS- SLS: A Service Level Negotiation Protocol for the Internet", ,2002

[91] Zeltserman, D. A Pratical Guide to SNMPv3 and Network Management, Prentice Hall, 1999

[92] Perkins, D., McGinnis, E., Understanding SNMP MIBs, Prentice Hall, 1997

[93] Harler, C., Web-Based Network Management, Wiley, 1999

[94] BIND 9 Administrator Reference Manual, 2001, Internet Software Consortium

[95] Langfeldt, Nicolai et al, DNS Howto v9, Dec 2001

[96] Johner, Heinz et al, Understanding LDAP. IBM Redbooks

[97] Levanen, Harri et al, Using LDAP for Directory Integration, IBM Redbooks

[98] System Administration Guide: Solaris Containers—Resource Management and Solaris Zones

[99] Bind 9 Administrator Reference Manual, Internet Software Consortium

[100] Ishiguro, K et al, Quagga – A routing software package for TCP/IP Networks, Dec. 2004

[101] Hubert, Bert, Linux Advanced Routing & Traffic Control How- to (iproute2), Linux Documentation Project, 2002

[102] Singh, Amit, An Introduction of Virtualization, 2005, www.*kernel*thread.com

[103] Kallahalla, M. et al, SoftUDC: A Software-Based Centro de dados for Utility Computing, Nov 2004, IEEE

[104] VMware GSX Administration Manual 3, www.vmware.com

[105] VMware GSX Virtual Machine Manual 3, www.vmware.com

[106] VMware GSX Guest OS Guide, www.vmware.com

[107] VMware GSX Server 3 Specifications, www.vmware.com

[108] Brian,Clark, et al, Xen and the Art of Repeated Research, Clarkson University

[109] The Xen Team, Xen interface manual, University of Cambridge, 2004

[110] The Xen Team, Xen User manual, University of Cambridge, 2004

[111] Xensource, Xensource Flyer

[112] CrowCroft, Jon, et al, The Inevitability of Xen, University of Cambridge, May 2005

[113] Rodriguez, Harold, Introduction to BASH shell scripting: Version 1.2, www.linuxorbit.com

[114] Cooper, M, Advanced Bash-Scripting Guide, Linux Documentation Project, May 2005

[115] Ritchie & Kernighan, The C Programming Language, Second Edition., Prentice Hall

[116] Husain, Kamran et al, Slackware Linux Unleashed 3rd Edition, Sams, 2002

[117] Mahadevan, I, Architecture and Experimental Plataforma for Supporting Qos in Wireless Networks Using Differential Services, ICCN '99, Boston, Oct 1999

[118] Salsano, S et al, QoS Control by Means of COPS to Support SIP- Based Applications, Universitá di Roma "Tor Vergata", IEEE Network March/April 2002

[119] Santos, A., Dias, B., Internet Network Services Management Plataforma, Departamento de Informática, Universidade do Minho, Oct 2003

[120] Phanse, K. et al, Design and Demostration of Policy Based Management in a Multi-Hop Ad Hoc Network, Virginia Polytechnic Institute and State University

[121] Zhuang, W. et al, Policy-Based QoS Management Architecture in an Integrated UMTS and WLAN Environment, Siemens Singapore, IEEE Communications Magazine, Nov 2003

6.3 Internet

[122] IETF Official Site, www.ietf.org

[123] DMTF Official Site, www.dmtf.org

[124] QEMU Emulator, www.qemu.org

[125] XenSource, www.xensource.com

[126] VMware Corporation, www.vmware.com

[127] Intel(R) COPS (Common Open Policy Service) Client Software Development Kit, Ver. 3.1, http://www.intel.com

[128] Suse Linux, http://www.novell.com/linux/suse/

[129] Microsoft Virtual Server, http://www.microsoft.com/windowsserversystem/virtualserver/

[130] Slackware Linux, www.slackware.org

[131] Vovida COPS Implementation, http://www.vovida.org/protocols/downloads/cops/

[132] Freesco Linux *Router*, www.freesco.org

[133] AdRem NetCrunch, www.adremsoft.com/netcrunch/

[134] Softerra LDAP Administrator, http://www.ldapadministrator.com/

[135] Tcl/tk Wikipedia, http://wiki.tcl.tk/

[136] Tcl Developer Xchange!, http://www.tcl.tk/

[137] OpenLDAP, http://www.openldap.org/

[138] Sun Solaris 10 documentation, http://www.sun.com/software/solaris/

[139] CRC 2005 - Conferência sobre Redes de Computadores, http://www.fccn.pt/crc2005/index.html

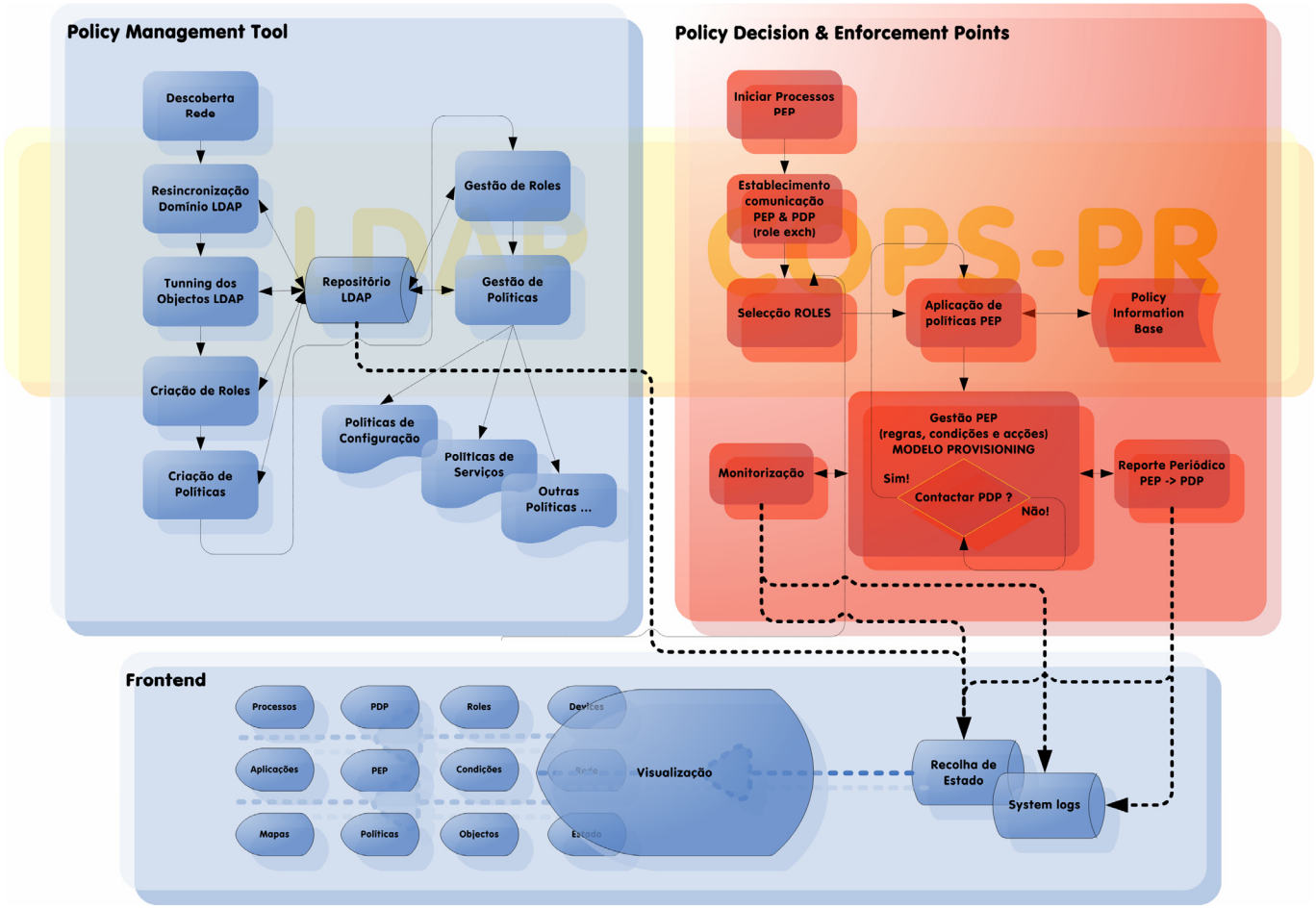
[140] Computer Communications Group - http://marco.uminho.pt/CCG/ccom.html

[141] Microsoft SAP Technology- http://www.microsoft-sap.com/technology.aspx

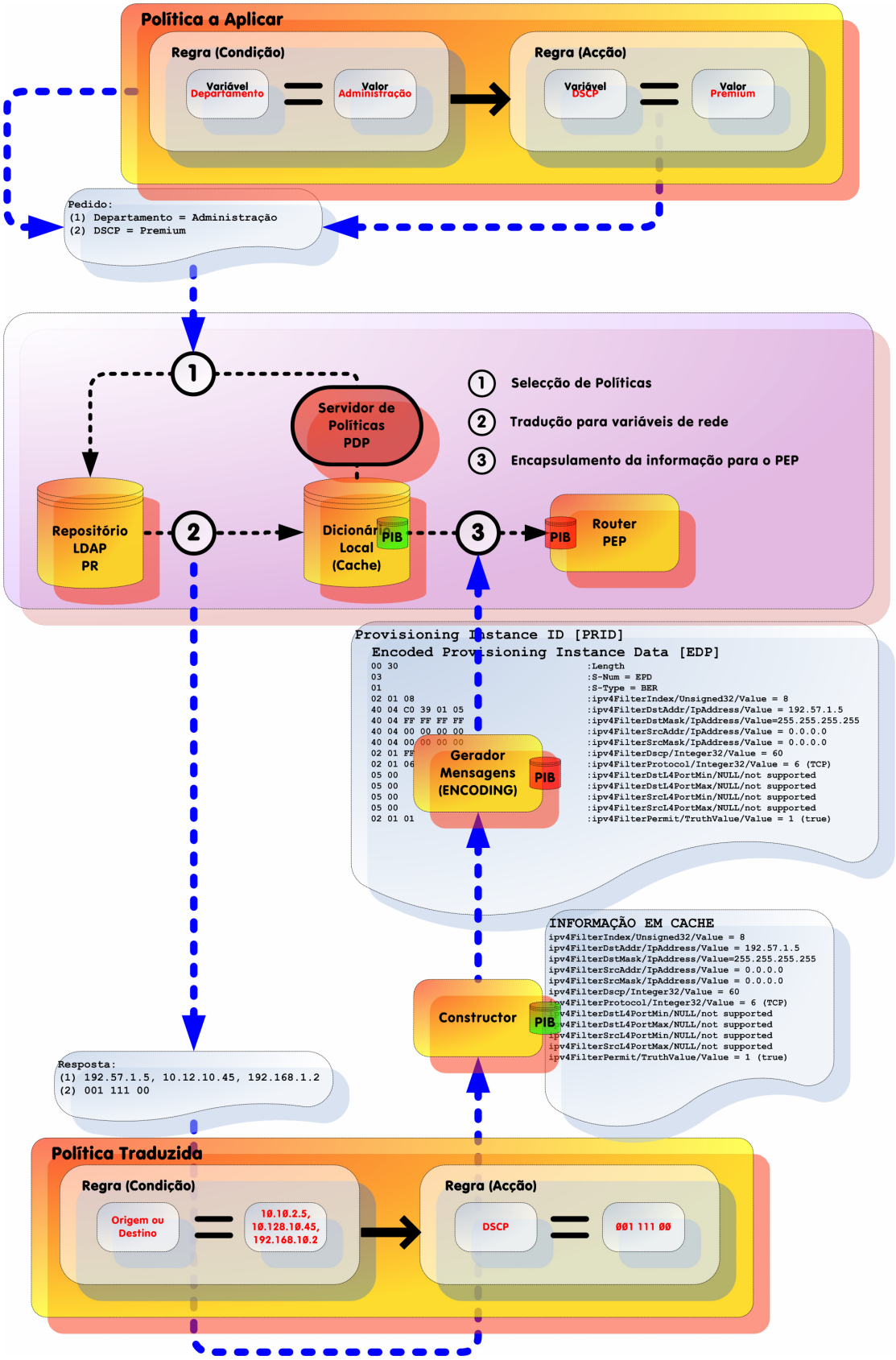
7 Anexos

7.1 Arquitectura CORDENA

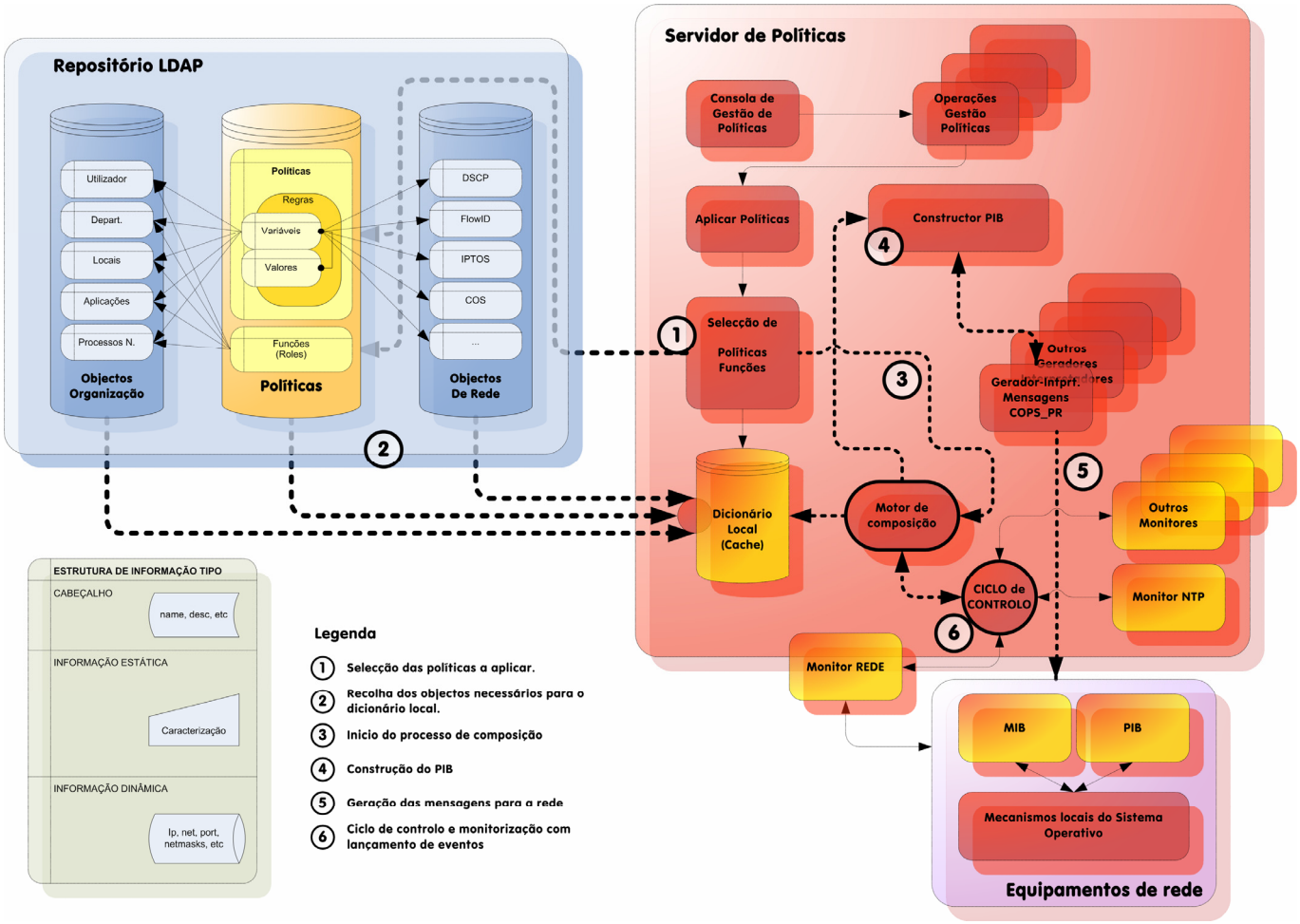
7.1.1 Fluxo de Processos e Operações



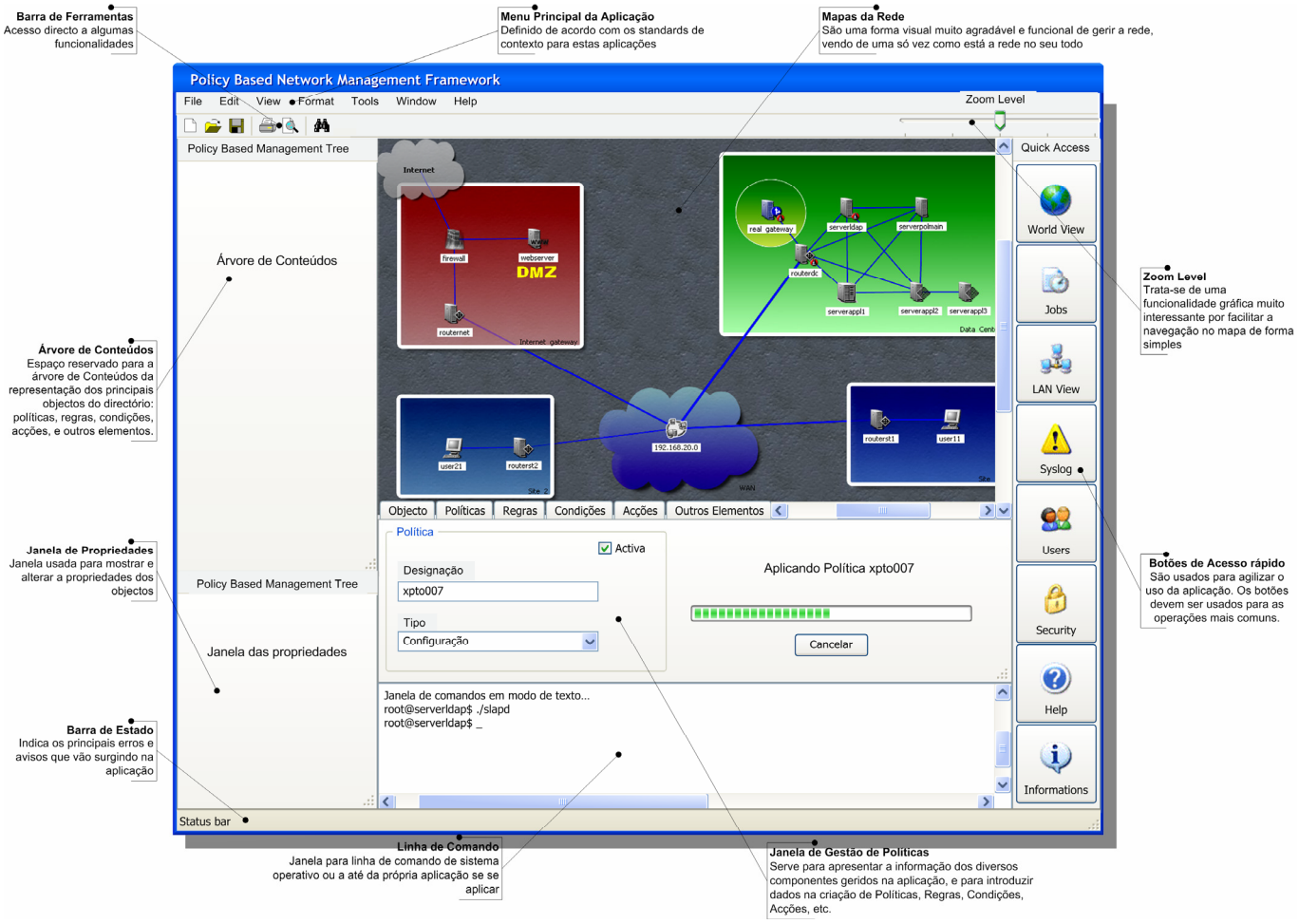
7.1.2 Mecanismos de Tradução de Políticas



7.1.3 Arquitectura Geral



7.1.4 Protótipo Gráfico da interface



7.2 OpenLDAP - *schema*s e configuração

7.2.1 Configuração slapd.conf

```
#
# See slapd.conf(5) for details on configuration options.
# This file should NOT be world readable.
# Guilherme: Com alterações respeitantes ao ambiente CORDENA - Ago 2005.
#
include  /usr/local/etc/openldap/schema /core.schema
include  /usr/local/etc/openldap/schema /cosine.schema
include  /usr/local/etc/openldap/schema /inetorgperson.schema
include  /usr/local/etc/openldap/schema /nis.schema
include  /usr/local/etc/openldap/schema /cordenacim25.schema
include  /usr/local/etc/openldap/schema /cordenapcim.schema
include  /usr/local/etc/openldap/schema /cordenapcime.schema

# Define global ACLs to disable default read access.

# Do not enable referrals until AFTER you have a working directory
# service AND an understanding of referrals.
# referral  ldap://root.openldap.org

loglevel  296
pidfile  /usr/local/var/run/slapd.pid
argsfile  /usr/local/var/run/slapd.args

# Load dynamic backend modules:
# modulepath/usr/local/libexec/openldap
# moduleloadback_bdb.la
# moduleloadback_ldap.la
# moduleloadback_ldbm.la
```

```
# moduleloadback_passwd.la
# moduleloadback_shell.la

# Sample security restrictions
# Require integrity protection (prevent hijacking)
# Require 112-bit (3DES or better) encryption for updates
# Require 63-bit encryption for simple bind
# security ssf=1 update_ssf=112 simple_bind=64

# Sample access control policy:
# Root DSE: allow anyone to read it
# Subschema  (sub)entry DSE: allow anyone to read it
# Other DSEs:
#   Allow self write access
#   Allow authenticated users read access
#   Allow anonymous users to authenticate
# Directives needed to implement policy:
# access to dn.base="" by * read
# access to dn.base="cn=Subschema " by * read
# access to *
# by self write
# by users read
# by anonymous auth
#
# if no access controls are present, the default policy
# allows anyone and everyone to read anything but restricts
# updates to rootdn.  (e.g., "access to * by * read")
#
# rootdn can always read and write EVERYTHING!

# Guilherne 16-10-2005: Retro-compatibilidade está fora do scope. Não é
suportado v.2.
require LDAPv3
```

```
## Misc security settings
password-hash  {SSHA}

#######################################################################
# BDB database definitions
#######################################################################

database  bdb
passwd
suffix   "dc=vnet,dc=lab"
rootdn   "cn=Manager,dc=vnet,dc=lab"
# Cleartext passwords, especially for the rootdn, should
# be avoid.  See slappasswd(8) and slapd.conf(5) for details.
# Use of strong authentication encouraged.
rootpw {SSHA}fIoFdFBgtkY4toFCOtb5m0l8V0Nn5cY+

# Para registar atributos de gestão, ex. última actualização das entra-
das (do core.schema )
lastmod on

# Para desactivar por completo (mesmo para o rootdn) acesso de escrita.
(Apenas leitura)
# readonly on

# The database directory MUST exist prior to running slapd AND
# should only be accessible by the slapd and slap tools.
# Mode 700 recommended.
directory /usr/local/var/openldap-data

# Indices to maintain
index cn,sn,uid pres,eq,approx,sub
index  objectClass eq
```

```
## db tuning parameters; cache 2000 entries in memory
# Guilherme: vigiar consumo de ram!
cachesize 2000
```

7.2.2 CORDENA DMTF CIM v2.5 *Schema*

```
# DMTF CIM 2.5 LDAP Schema  - Writen by Guilherme Teixeira 2005-10-15

attributetype ( 1.3.6.1.4.1.412.100.1.2.5 NAME 'arrayIndex'
  DESC 'The index of this child.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.101 NAME 'dlmIdentifyingDescrip-
tion'
  DESC 'A free-form string providing explanation and
  details behind the entries in the dlmOtherIdentifyingInfo
  attribute.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.112 NAME 'dlmOtherIdentify-
ingInfo'
  DESC 'OtherIdentifyingInfo captures additional data,
  beyond that of Tag information, that could be used to
  identify a Physical Element. One example is bar code
  data associated with an Element that also has an asset
  tag. Note that if only bar code data is available and
  is unique/able to be used as an Element key, this
  property would be NULL and the bar code data used as
  the class key, in the Tag property.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE
  EQUALITY caseIgnoreMatch)
```

```
objectclass ( 1.3.6.1.4.1.412.100.2.1.3.92
  NAME 'dlmOtherIdentifyingInfoInstance'
  DESC 'Helper class to tie indexed arrays in Core Model together.'
  SUP top
  MUST ( arrayIndex )
  MAY ( dlmOtherIdentifyingInfo $ dlmIdentifyingDescription ))

#objectclass ( 1.3.6.1.4.1.412.100.2.3.3.9
# NAME 'dlmOtherIdentifyingInfoInstanceNameForm'
# OC dlmOtherIdentifyingInfoInstance
# MUST ( arrayIndex ))

#( <core-sr-9> NAME 'dlmOtherIdentifyingInfoInstanceStructureRule'
# FORM dlmOtherIdentifyingInfoInstanceNameForm
#)

attributetype ( 1.3.6.1.4.1.412.100.1.2.1 NAME 'orderedCimKeys'
  DESC 'The model path for the instance. May be used as an RDN.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.1.2.2 NAME 'orderedCimModelPath'
  DESC 'The model path for the instance (with propagated keys). May
  be used as an RDN.'
  OBSOLETE
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.103 NAME 'dlmCaption'
  DESC 'The Caption property is a short textual
  description (oneline string) of the object.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
```

```
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.104 NAME 'dlmDescription'
  DESC 'The Description property provides a textual
  description of the object.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

# ### dlm1ManagedElement ###
objectclass ( 1.3.6.1.4.1.412.100.2.1.3.1 NAME 'dlm1ManagedElement'
  DESC 'ManagedElement is an abstract class that provides
  a common superclass (or top of the inheritance tree)
  for the non-association classes in the CIM Schema .'
  SUP top ABSTRACT
  MAY ( dlmCaption $ dlmDescription $ orderedCimModelPath
  $ orderedCimKeys ))

attributetype ( 1.3.6.1.4.1.412.100.2.2.105 NAME 'dlmInstallDate'
  DESC 'A datetime value indicating when the object was
  installed. A lack of a value does not indicate that
  the object is not installed.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE
  EQUALITY generalizedTimeMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.106 NAME 'dlmName'
  DESC 'The Name property defines the label by which the
  object is known. When subclassed, the Name property
  can be overridden to be a Key property.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.107 NAME 'dlmStatus'
  DESC 'A string indicating the current status of the
```

object. Various operational and non-operational statuses are defined. Operational statuses are "OK", "Degraded", "Stressed" and "Pred Fail". "Stressed" indicates that the Element is functioning, but needs attention. Examples of "Stressed" states are overload, overheated, etc. The condition "Pred Fail" (failure predicted) indicates that an Element is functioning properly but predicting a failure in the near future. An example is a SMART-enabled hard drive.
Non-operational statuses can also be specified. These are "Error", "NonRecover", "Starting", "Stopping", "Stopped", "Service", "No Contact" and "Lost Comm". "NonRecover" indicates that a non-recoverable error has occurred. "Service" describes an Element being configured, maintained, cleaned, or otherwise administered. This status could apply during mirror-resilvering of a disk, reload of a user permissions list, or other administrative task. Not all such work is on-line, yet the Element is neither "OK" nor in one of the other states. "No Contact" indicates that the current instance of the monitoring system has knowledge of this Element but has never been able to establish communications with it. "Lost Comm" indicates that the ManagedSystemElement is known to exist and has been contacted successfully in the past, but is currently unreachable."Stopped" indicates that the ManagedSystemElement is known to exist, it is not operational (i.e. it is unable to provide service to users), but it has not failed. It has purposely been made non-operational. The Element may have never been "OK", the Element may have initiated its own stop, or a management system may have initiated the stop. Value Mappings are "OK", "Error", "Degraded",

```
  "Unknown", "Pred Fail", "Starting", "Stopping",
  "Service", "Stressed", "NonRecover", "No Contact",
  "Lost Comm", "Stopped".'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{10} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

# ### dlm1ManagedSystemElement ###
objectclass ( 1.3.6.1.4.1.412.100.2.1.3.2 NAME
'dlm1ManagedSystemElement'
  DESC 'ManagedSystemElement is the base class for the
  System Element hierarchy. Membership Criteria: Any
  distinguishable component of a System is a candidate
  for inclusion in this class. Examples: software
  components, such as files; and devices, such as disk
  drives and controllers, and physical components such
  as chips and cards.'
  SUP dlm1ManagedElement ABSTRACT
  MAY ( dlmInstallDate $ dlmName $ dlmStatus ))

attributetype ( 1.3.6.1.4.1.412.100.2.2.108 NAME 'dlmCreationClassName'
  DESC 'CreationClassName indicates the name of the class
  or the subclass used in the creation of an instance.
  When used with the other key properties of this class,
  this property allows all instances of this class and
  its subclasses to be uniquely identified.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.109 NAME 'dlmManufactureDate'
  DESC 'Date that this PhysicalElement was manufactured.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.24 SINGLE-VALUE
  EQUALITY generalizedTimeMatch)
```

```
attributetype ( 1.3.6.1.4.1.412.100.2.2.110 NAME 'dlmManufacturer'
  DESC 'The name of the organization responsible for
  producing the PhysicalElement. This may be the entity
  from whom the Element is purchased, but this is not
  necessarily true. The latter information is contained
  in the Vendor property of Product.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.111 NAME 'dlmModel'
  DESC 'The name by which the PhysicalElement is
  generally known.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.113 NAME 'dlmPartNumber'
  DESC 'The part number assigned by the organization
  responsible for producing or manufacturing the
  PhysicalElement.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.114 NAME 'dlmPoweredOn'
  DESC 'Boolean indicating that the PhysicalElement is
  powered on (TRUE), or is currently off (FALSE).'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.7 SINGLE-VALUE)

attributetype ( 1.3.6.1.4.1.412.100.2.2.115 NAME 'dlmSKU'
  DESC 'The stock keeping unit number for this
  PhysicalElement.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)
```

```
attributetype ( 1.3.6.1.4.1.412.100.2.2.116 NAME 'dlmSerialNumber'
  DESC 'A manufacturer-allocated number used to identify
  the Physical Element.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.117 NAME 'dlmTag'
  DESC 'An arbitrary string that uniquely identifies the
  Physical Element and serves as the Element"s key. The
  Tag property can contain information such as asset tag
  or serial number data. The key for PhysicalElement is
  placed very high in the object hierarchy in order to
  independently identify the hardware/entity, regardless
  of physical placement in or on Cabinets, Adapters, etc.
  For example, a hotswappable or removeable component
  may be taken from its containing (scoping) Package and
  be temporarily unused. The object still continues to
  exist - and may even be inserted into a different
  scoping container. Therefore, the key for Physical
  Element is an arbitrary string and is defined
  independently of any placement or location-oriented
  hierarchy.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.118 NAME 'dlmVersion'
  DESC 'A string indicating the version of the
  PhysicalElement.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

objectclass ( 1.3.6.1.4.1.412.100.2.1.3.3 NAME 'dlm1PhysicalElement'
  DESC 'Subclasses of PhysicalElement define any
```

```
  component of a System that has a distinct physical
  identity. Instances of this class can be defined in
  terms of labels that can be physically attached to the
  object. All Processes, Files, and LogicalDevices are
  not considered to be Physical Elements. For example,
  it is not possible to attach a label to a modem. It is
  only possible to attach a label to the card that
  implements the modem. The same card could also
  implement a LAN adapter. These are tangible Managed
  System Elements (usually actual hardware items) that
  have a physical manifestation of some sort. A Managed
  System Element is not necessarily a discrete
  component. For example, it is possible for a single
  Card (which is a type of Physical Element) to host
  more than one Logical Device. The card would be
  represented by a single Physical Element associated
  with multiple Logical Devices.'
  SUP dlm1ManagedSystemElement ABSTRACT
  MAY ( dlmCreationClassName $ dlmManufactureDate $
  dlmManufacturer $ dlmModel $ dlmOtherIdentifyingInfo $
  dlmPartNumber $ dlmPoweredOn $ dlmSKU $ dlmSerialNumber $
  dlmTag $ dlmVersion ))

objectclass ( 1.3.6.1.4.1.412.100.2.1.3.4 NAME 'dlm1LogicalElement'
  DESC 'LogicalElement is a base class for all the
  components of a System that represent logical
  entities, such as Files, Processes, and
  Logical Devices.'
  SUP dlm1ManagedSystemElement ABSTRACT)

attributetype ( 1.3.6.1.4.1.412.100.2.2.119 NAME 'dlmNameFormat'
  DESC 'The System object and its derivatives are Top
  Level Objects of CIM. They provide the scope for
```

```
  numerous components. Having unique System keys is
  required. A heuristic can be defined in individual
  System subclasses to attempt to always generate the
  same System Name Key. The NameFormat property
  identifies how the System name was generated, using
  the subclass" heuristic.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.120 NAME 'dlmPrimaryOwnerCon-
tact'
  DESC 'A string that provides information on how the
  primary system owner can be reached (e.g. phone
  number, email address, ...).'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{256} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.121 NAME 'dlmPrimaryOwnerName'
  DESC 'The name of the primary system owner.'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} SINGLE-VALUE
  EQUALITY caseIgnoreMatch)

attributetype ( 1.3.6.1.4.1.412.100.2.2.122 NAME 'dlmroles'
  DESC 'An array (bag) of strings that specify the roles
  this System plays in the IT-environment. Subclasses of
  System may override this property to define explicit
  roles values. Alternately, a Working Group may
  describe the heuristics, conventions and guidelines
  for specifying roles. For example, for an instance of
  a networking system, the roles property might contain
  the string, "Switch" or "Bridge".'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  EQUALITY caseIgnoreMatch)
```

```
# ### dlm1System ###
objectclass ( 1.3.6.1.4.1.412.100.2.1.3.5 NAME 'dlm1System'
  DESC 'A System is a LogicalElement that aggregates an
  enumerable set of Managed System Elements. The
  aggregation operates as a functional whole. Within any
  particular subclass of System, there is a well-defined
  list of Managed System Element classes whose instances
  must be aggregated.'
  SUP dlm1LogicalElement ABSTRACT
  MAY ( dlmCreationClassName $ dlmName $ dlmNameFormat $
  dlmPrimaryOwnerContact $ dlmPrimaryOwnerName $
  dlmroles ))

attributetype ( 1.3.6.1.4.1.412.100.2.2.123 NAME 'dlmDedicated'
  DESC 'Enumeration indicating whether the ComputerSystem
  is a special-purpose System (ie, dedicated to a
  particular use), versus being "general purpose". For
  example, one could specify that the System is
  dedicated to "Print" (value=11) or acts as a "Hub"
  (value=8). Values are 0="Not Dedicated",
  1="Unknown", 2="Other", 3="Storage", 4="Router",
  5="Switch", 6="Layer 3 Switch", 7="Central Office
  Switch", 8="Hub", 9="Access Server", 10="Firewall",
  11="Print", 12="I/O", 13="Web Caching",
  14="Management"'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
  EQUALITY integerMatch)

objectclass ( 1.3.6.1.4.1.412.100.2.1.3.6 NAME 'dlm1ComputerSystem'
  DESC 'A class derived from System that is a special
  collection of ManagedSystemElements. This collection
  provides compute capabilities and serves as
```

```
  aggregation point to associate one or more of the
  following elements: FileSystem, OperatingSystem,
  Processor and Memory (Volatile and/or NonVolatile
  Storage).'
  SUP dlm1System ABSTRACT
  MAY ( dlmDedicated $ dlmNameFormat ))

objectclass ( 1.3.6.1.4.1.412.100.2.1.3.93 NAME 'dlm1AdminDomain'
  DESC 'This is a special grouping of
  ManagedSystemElements. The grouping is viewed as a
  single entity, reflecting that all of its components
  are administered similarly - either by the same user,
  group of users or policy. It serves as an aggregation
  point to associate one or more of the following
  elements: network devices, such as routers and
  switches, servers, and other resources that can be
  accessed by end systems. This grouping of devices
  plays an essential role in ensuring that the same
  administrative policy and actions are applied to all
  of the devices in the grouping. The specific behavior
  and/or semantics of the AdminDomain can be identified
  through its aggregated and associated entities.'
  SUP dlm1System ABSTRACT)

objectclass ( 1.3.6.1.4.1.412.100.2.1.3.94 NAME
'dlm1AdminDomainAuxClass'
  DESC 'This is a special grouping of
  ManagedSystemElements. The grouping is viewed as a
  single entity, reflecting that all of its components
  are administered similarly - either by the same user,
  group of users or policy. It serves as an aggregation
  point to associate one or more of the following
  elements: network devices, such as routers and
```

```
  switches, servers, and other resources that can be
  accessed by end systems. This grouping of devices
  plays an essential role in ensuring that the same
  administrative policy and actions are applied to all
  of the devices in the grouping. The specific behavior
  and/or semantics of the AdminDomain can be identified
  through its aggregated and associated entities.'
  SUP dlm1AdminDomain AUXILIARY)

## Implementa as 5 classes necessárias do doc DSP0123.pdf do DMTF
```

7.2.3 CORDENA PCIM *Schema*

```
# Schema  file created by Guilherme Teixeira - 2005-10-15
# Based on RFC 3703

attributetype ( 1.3.6.1.1.6.2.3 NAME 'pcimKeywords'
              DESC 'A set of keywords to assist directory clients in
                    locating the policy objects applicable to them.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
       )

objectclass ( 1.3.6.1.1.6.1.1 NAME 'pcimPolicy'
         DESC 'An abstract class that is the base class for all classes
               that describe policy-related instances.'
         SUP dlm1ManagedElement
         ABSTRACT
         MAY ( cn $ dlmCaption $ dlmDescription $ orderedCimKeys $
               pcimKeywords )
       )
```

```
attributetype ( 1.3.6.1.1.6.2.4 NAME 'pcimGroupName'
              DESC 'The user-friendly name of this policy group.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
       )

objectclass ( 1.3.6.1.1.6.1.2 NAME 'pcimGroup'
              DESC 'A container for a set of related pcimRules and/or
                    a set of related pcimGroups.'
              SUP pcimPolicy
              ABSTRACT
              MAY ( pcimGroupName )
       )


objectclass ( 1.3.6.1.1.6.1.3 NAME 'pcimGroupAuxClass'
              DESC 'An auxiliary class that collects a set of related
                    pcimRule and/or pcimGroup entries.'
              SUP pcimGroup
              AUXILIARY
       )

objectclass ( 1.3.6.1.1.6.1.4 NAME 'pcimGroupInstance'
              DESC 'A structural class that collects a set of related
                    pcimRule and/or pcimGroup entries.'
              SUP pcimGroup
              STRUCTURAL
       )
```

```
attributetype ( 1.3.6.1.1.6.2.5 NAME 'pcimRuleName'
                DESC 'The user-friendly name of this policy rule.'
                EQUALITY caseIgnoreMatch
                ORDERING caseIgnoreOrderingMatch
                SUBSTR caseIgnoreSubstringsMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
                SINGLE-VALUE
        )

attributetype ( 1.3.6.1.1.6.2.6 NAME 'pcimRuleEnabled'
                DESC 'An integer indicating whether a policy rule is
                      administratively enabled (value=1), disabled
                      (value=2), or enabled for debug (value=3).'
                EQUALITY integerMatch
                ORDERING integerOrderingMatch
                SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
                SINGLE-VALUE
        )

attributetype ( 1.3.6.1.1.6.2.7 NAME 'pcimRuleConditionListType'
            DESC 'A value of 1 means that this policy rule is in
                  disjunctive normal form; a value of 2 means that this
                  policy rule is in conjunctive normal form.'
            EQUALITY integerMatch
            ORDERING integerOrderingMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
            SINGLE-VALUE
     )

attributetype ( 1.3.6.1.1.6.2.8 NAME 'pcimRuleConditionList'
            DESC 'Unordered set of DNs of pcimRuleConditionAssociation
                  entries representing associations between this policy
                  rule and its conditions.'
```

```
            EQUALITY distinguishedNameMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     )

attributetype ( 1.3.6.1.1.6.2.9 NAME 'pcimRuleActionList'
            DESC 'Unordered set of DNs of pcimRuleActionAssociation
                  entries representing associations between this policy
                  rule and its actions.'
           EQUALITY distinguishedNameMatch
           SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     )

attributetype ( 1.3.6.1.1.6.2.10 NAME 'pcimRuleValidityPeriodList'
            DESC 'Unordered set of DNs of pcimRuleValidityAssociation
                  entries that determine when the pcimRule is scheduled
                  to be active or inactive.'
            EQUALITY distinguishedNameMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     )

attributetype ( 1.3.6.1.1.6.2.11 NAME 'pcimRuleUsage'
            DESC 'This attribute is a free-form sting providing
                  guidelines on how this policy should be used.'
            EQUALITY caseIgnoreMatch
            ORDERING caseIgnoreOrderingMatch
            SUBSTR caseIgnoreSubstringsMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
            SINGLE-VALUE
     )

attributetype ( 1.3.6.1.1.6.2.12 NAME 'pcimRulePriority'
            DESC 'A non-negative integer for prioritizing this
                  pcimRule relative to other pcimRules.  A larger
```

```
                value indicates a higher priority.'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE
    )

attributetype ( 1.3.6.1.1.6.2.13 NAME 'pcimRuleMandatory'
        DESC 'If TRUE, indicates that for this policy rule, the
                evaluation of its conditions and execution of its
                actions (if the condition is satisfied) is required.'
        EQUALITY booleanMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        SINGLE-VALUE
    )

attributetype ( 1.3.6.1.1.6.2.14 NAME 'pcimRuleSequencedActions'
        DESC 'An integer enumeration indicating that the ordering of
                actions defined by the pcimActionOrder attribute is
                mandatory(1), recommended(2), or dontCare(3).'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE
    )

attributetype ( 1.3.6.1.1.6.2.15 NAME 'pcim roles'
        DESC 'Each value of this attribute represents a role-
                combination.'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
```

```
     )

objectclass ( 1.3.6.1.1.6.1.5 NAME 'pcimRule'
            DESC 'The base class for representing the "If Condition
                  then Action" semantics associated with a policy rule.'
            SUP pcimPolicy
            ABSTRACT
            MAY ( pcimRuleName $ pcimRuleEnabled $
                  pcimRuleConditionListType $ pcimRuleConditionList $
                  pcimRuleActionList $ pcimRuleValidityPeriodList $
                  pcimRuleUsage $ pcimRulePriority $
                  pcimRuleMandatory $ pcimRuleSequencedActions $
                  pcim roles )
     )

objectclass ( 1.3.6.1.1.6.1.6 NAME 'pcimRuleAuxClass'
            DESC 'An auxiliary class for representing the "If Condition
                 then Action" semantics associated with a policy rule.'
            SUP pcimRule
            AUXILIARY
     )

objectclass ( 1.3.6.1.1.6.1.7 NAME 'pcimRuleInstance'
            DESC 'A structural class for representing the "If Condition
                 then Action" semantics associated with a policy rule.'
            SUP pcimRule
            STRUCTURAL
     )


attributetype ( 1.3.6.1.1.6.2.16
            NAME 'pcimConditionGroupNumber'
            DESC 'The number of the group to which a policy condition
```

```
                belongs.  This is used to form the DNF or CNF
                expression associated with a policy rule.'
        EQUALITY integerMatch
        ORDERING integerOrderingMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
        SINGLE-VALUE
    )

attributetype ( 1.3.6.1.1.6.2.17
        NAME 'pcimConditionNegated'
        DESC 'If TRUE (FALSE), it indicates that a policy condition
                IS (IS NOT) negated in the DNF or CNF expression
                associated with a policy rule.'
        EQUALITY booleanMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        SINGLE-VALUE
    )

attributetype ( 1.3.6.1.1.6.2.18
        NAME 'pcimConditionName'
        DESC 'A user-friendly name for a policy condition.'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE
    )

attributetype ( 1.3.6.1.1.6.2.19
        NAME 'pcimConditionDN'
        DESC 'A DN that references an instance of a reusable policy
                condition.'
        EQUALITY distinguishedNameMatch
```

```
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
            SINGLE-VALUE
     )

objectclass ( 1.3.6.1.1.6.1.8 NAME 'pcimRuleConditionAssociation'
            DESC 'This class contains attributes characterizing the
                  relationship between a policy rule and one of its
                  policy conditions.'
            SUP pcimPolicy
            MUST ( pcimConditionGroupNumber $ pcimConditionNegated )
            MAY ( pcimConditionName $ pcimConditionDN )
     )


attributetype ( 1.3.6.1.1.6.2.20
            NAME 'pcimValidityConditionName'
            DESC 'A user-friendly name for identifying an instance of
                  a pcimRuleValidityAssociation entry.'
            EQUALITY caseIgnoreMatch
            ORDERING caseIgnoreOrderingMatch
            SUBSTR caseIgnoreSubstringsMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
            SINGLE-VALUE
     )

attributetype ( 1.3.6.1.1.6.2.21
            NAME 'pcimTimePeriodConditionDN'
             DESC 'A reference to a reusable policy time period
                   condition.'
            EQUALITY distinguishedNameMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
            SINGLE-VALUE
     )
```

```
objectclass ( 1.3.6.1.1.6.1.9 NAME 'pcimRuleValidityAssociation'
           DESC 'This defines the scheduled activation or deactivation
                 of a policy rule.'
           SUP pcimPolicy
           STRUCTURAL
           MAY ( pcimValidityConditionName $ pcimTimePeriodConditionDN )
     )


attributetype ( 1.3.6.1.1.6.2.22
            NAME 'pcimActionName'
            DESC 'A user-friendly name for a policy action.'
            EQUALITY caseIgnoreMatch
            ORDERING caseIgnoreOrderingMatch
            SUBSTR caseIgnoreSubstringsMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
            SINGLE-VALUE
     )

attributetype ( 1.3.6.1.1.6.2.23
            NAME 'pcimActionOrder'
            DESC 'An integer indicating the relative order of an action
                  in the context of a policy rule.'
            EQUALITY integerMatch
            ORDERING integerOrderingMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
            SINGLE-VALUE
     )

attributetype ( 1.3.6.1.1.6.2.24
            NAME 'pcimActionDN'
            DESC 'A DN that references a reusable policy action.'
```

```
            EQUALITY distinguishedNameMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
            SINGLE-VALUE
     )

objectclass ( 1.3.6.1.1.6.1.10 NAME 'pcimRuleActionAssociation'
            DESC 'This class contains attributes characterizing the
                  relationship between a policy rule and one of its
                  policy actions.'
            SUP pcimPolicy
            MUST ( pcimActionOrder )
            MAY ( pcimActionName $ pcimActionDN )
     )

objectclass ( 1.3.6.1.1.6.1.11 NAME 'pcimConditionAuxClass'
            DESC 'A class representing a condition to be evaluated in
                  conjunction with a policy rule.'
            SUP top
            AUXILIARY
     )


attributetype ( 1.3.6.1.1.6.2.25
               NAME 'pcimTPCTime'
               DESC 'The start and end times on which a policy rule is
                     valid.'
               EQUALITY caseIgnoreMatch
               ORDERING caseIgnoreOrderingMatch
               SUBSTR caseIgnoreSubstringsMatch
               SYNTAX 1.3.6.1.4.1.1466.115.121.1.44
               SINGLE-VALUE
        )
```

```
attributetype ( 1.3.6.1.1.6.2.26
            NAME 'pcimTPCMonthOfYearMask'
            DESC 'This identifies the valid months of the year for a
                  policy rule using a 12-bit string that represents the
                  months of the year from January through December.'
            EQUALITY bitStringMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.6
            SINGLE-VALUE
     )

attributetype ( 1.3.6.1.1.6.2.27
            NAME 'pcimTPCDayOfMonthMask'
            DESC 'This identifies the valid days of the month for a
                  policy rule using a 62-bit string. The first 31
                  positions represent the days of the month in ascending
                  order, and the next 31 positions represent the days of
                  the month in descending order.'
            EQUALITY bitStringMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.6
            SINGLE-VALUE
     )

attributetype ( 1.3.6.1.1.6.2.28
            NAME 'pcimTPCDayOfWeekMask'
            DESC 'This identifies the valid days of the week for a
                  policy rule using a 7-bit string. This represents
                  the days of the week from Sunday through Saturday.'
            EQUALITY bitStringMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.6
            SINGLE-VALUE
     )

attributetype ( 1.3.6.1.1.6.2.29
```

```
            NAME 'pcimTPCTimeOfDayMask'
            DESC 'This identifies the valid range of times for a policy
                  using the format Thhmmss/Thhmmss.'
            EQUALITY caseIgnoreMatch
            ORDERING caseIgnoreOrderingMatch
            SUBSTR caseIgnoreSubstringsMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.44
            SINGLE-VALUE
     )

attributetype ( 1.3.6.1.1.6.2.30
            NAME 'pcimTPCLocalOrUtcTime'
            DESC 'This defines whether the times in this instance
                  represent local (value=1) times or UTC (value=2)
                  times.'
            EQUALITY integerMatch
            ORDERING integerOrderingMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
            SINGLE-VALUE
     )

objectclass ( 1.3.6.1.1.6.1.12 NAME 'pcimTPCAuxClass'
            DESC 'This provides the capability of enabling or disabling
                  a policy rule according to a predetermined schedule.'
            SUP pcimConditionAuxClass
            AUXILIARY
            MAY ( pcimTPCTime $ pcimTPCMonthOfYearMask $
                  pcimTPCDayOfMonthMask $ pcimTPCDayOfWeekMask $
                  pcimTPCTimeOfDayMask $ pcimTPCLocalOrUtcTime )
     )


attributetype ( 1.3.6.1.1.6.2.31
```

```
            NAME 'pcimVendorConstraintData'
            DESC 'Mechanism for representing constraints that have not
                  been modeled as specific attributes.  Their format is
                  identified by the OID stored in the attribute
                  pcimVendorConstraintEncoding.'
            EQUALITY octetStringMatch
            ORDERING octetStringOrderingMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
     )

attributetype ( 1.3.6.1.1.6.2.32
            NAME 'pcimVendorConstraintEncoding'
            DESC 'An OID identifying the format and semantics for the
                  pcimVendorConstraintData for this instance.'
            EQUALITY objectIdentifierMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
            SINGLE-VALUE
     )

objectclass ( 1.3.6.1.1.6.1.13 NAME 'pcimConditionVendorAuxClass'
            DESC 'A class that defines a registered means to describe a
                  policy condition.'
            SUP pcimConditionAuxClass
            AUXILIARY
            MAY ( pcimVendorConstraintData $
                 pcimVendorConstraintEncoding )
     )

objectclass ( 1.3.6.1.1.6.1.14 NAME 'pcimActionAuxClass'
            DESC 'A class representing an action to be performed as a
                  result of a policy rule.'
            SUP top
            AUXILIARY
```

```
     )

attributetype ( 1.3.6.1.1.6.2.33
            NAME 'pcimVendorActionData'
            DESC ' Mechanism for representing policy actions that have
                   not been modeled as specific attributes.  Their
                   format is identified by the OID stored in the
                   attribute pcimVendorActionEncoding.'
            EQUALITY octetStringMatch
            ORDERING octetStringOrderingMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
     )

attributetype ( 1.3.6.1.1.6.2.34
            NAME 'pcimVendorActionEncoding'
            DESC 'An OID identifying the format and semantics for the
                  pcimVendorActionData attribute of this instance.'
            EQUALITY objectIdentifierMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
            SINGLE-VALUE
     )

objectclass ( 1.3.6.1.1.6.1.15 NAME 'pcimActionVendorAuxClass'
            DESC 'A class that defines a registered means to describe a
                  policy action.'
            SUP pcimActionAuxClass
            AUXILIARY
            MAY ( pcimVendorActionData $ pcimVendorActionEncoding )
     )


attributetype ( 1.3.6.1.1.6.2.35 NAME 'pcimPolicyInstanceName'
            DESC 'The user-friendly name of this policy instance.'
```

```
            EQUALITY caseIgnoreMatch
            ORDERING caseIgnoreOrderingMatch
            SUBSTR caseIgnoreSubstringsMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
            SINGLE-VALUE
     )

objectclass ( 1.3.6.1.1.6.1.16 NAME 'pcimPolicyInstance'
            DESC 'A structural class to which aux classes containing
                  reusable policy information can be attached.'
            SUP pcimPolicy
            MAY ( pcimPolicyInstanceName )
     )

objectclass ( 1.3.6.1.1.6.1.17 NAME 'pcimElementAuxClass'
            DESC 'An auxiliary class used to tag instances of classes
                  defined outside the realm of policy as relevant to a
                  particular policy specification.'
            SUP pcimPolicy
            AUXILIARY
     )


attributetype ( 1.3.6.1.1.6.2.36 NAME 'pcimRepositoryName'
            DESC 'The user-friendly name of this Policy Repository.'
            EQUALITY caseIgnoreMatch
            ORDERING caseIgnoreOrderingMatch
            SUBSTR caseIgnoreSubstringsMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
            SINGLE-VALUE
     )

objectclass ( 1.3.6.1.1.6.1.18 NAME 'pcimRepository'
```

```
            DESC 'A container for reusable policy information.'
            SUP dlm1AdminDomain
            ABSTRACT
            MAY ( pcimRepositoryName )
     )

objectclass ( 1.3.6.1.1.6.1.19 NAME 'pcimRepositoryAuxClass'
            DESC 'An auxiliary class that can be used to aggregate
                  reusable policy information.'
            SUP pcimRepository
            AUXILIARY
     )

objectclass ( 1.3.6.1.1.6.1.20 NAME 'pcimRepositoryInstance'
            DESC 'A structural class that can be used to aggregate
                  reusable policy information.'
            SUP pcimRepository
            STRUCTURAL
     )

attributetype ( 1.3.6.1.1.6.2.37
            NAME 'pcimSubtreesAuxContainedSet'
            DESC 'DNs of objects that serve as roots for DIT subtrees
                  containing policy-related objects.'
            EQUALITY distinguishedNameMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     )

objectclass ( 1.3.6.1.1.6.1.21 NAME 'pcimSubtreesPtrAuxClass'
            DESC 'An auxiliary class providing DN references to roots of
                  DIT subtrees containing policy-related objects.'
            SUP top
            AUXILIARY
```

```
            MAY ( pcimSubtreesAuxContainedSet )
     )

attributetype ( 1.3.6.1.1.6.2.38
            NAME 'pcimGroupsAuxContainedSet'
            DESC 'DNs of pcimGroups associated in some way with the
                  instance to which this attribute has been appended.'
            EQUALITY distinguishedNameMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     )

objectclass ( 1.3.6.1.1.6.1.22 NAME 'pcimGroupContainmentAuxClass'
            DESC 'An auxiliary class used to bind pcimGroups to an
                  appropriate container object.'
            SUP top
            AUXILIARY
            MAY ( pcimGroupsAuxContainedSet )
     )

attributetype ( 1.3.6.1.1.6.2.39
            NAME 'pcimRulesAuxContainedSet'
            DESC 'DNs of pcimRules associated in some way with the
                  instance to which this attribute has been appended.'
            EQUALITY distinguishedNameMatch
            SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     )

objectclass ( 1.3.6.1.1.6.1.23 NAME 'pcimRuleContainmentAuxClass'
            DESC 'An auxiliary class used to bind pcimRules to an
                  appropriate container object.'
            SUP top
            AUXILIARY
            MAY ( pcimRulesAuxContainedSet )
```

```
    )
```

7.2.4 CORDENA PCIME *Schema*

```
# Schema  file 2005-10-15 Guilherme Teixeira
# Policy Core Extension LDAP Schema  as defined by RFC 4104


attributetype ( 1.3.6.1.1.9.2.1
     NAME 'pcelsPolicySetName'
     DESC 'User-friendly name of a policy set'
     EQUALITY caseIgnoreMatch
     ORDERING caseIgnoreOrderingMatch
     SUBSTR caseIgnoreSubstringsMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.2
     NAME 'pcelsDecisionStrategy'
     DESC 'Evaluation method for the components of a pcelsPolicySet'
     EQUALITY integerMatch
     ORDERING integerOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.3
     NAME 'pcelsPolicySetList'
     DESC 'Unordered set of DNs of pcelsPolicySetAssociation entries'
     EQUALITY distinguishedNameMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
   )
```

```
objectclass ( 1.3.6.1.1.9.1.1
     NAME 'pcelsPolicySet'
     DESC 'Set of policies'
     SUP pcimPolicy
     ABSTRACT
     MAY ( pcelsPolicySetName
         $ pcelsDecisionStrategy
         $ pcim roles
         $ pcelsPolicySetList )
   )

attributetype ( 1.3.6.1.1.9.2.4
     NAME 'pcelsPriority'
     DESC 'Priority of a component'
     EQUALITY integerMatch
     ORDERING integerOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.5
     NAME 'pcelsPolicySetDN'
     DESC 'DN of a pcelsPolicySet entry'
     EQUALITY distinguishedNameMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     SINGLE-VALUE
   )

objectclass ( 1.3.6.1.1.9.1.2
     NAME 'pcelsPolicySetAssociation'
     DESC 'Associates a policy set to an aggregating entry'
     SUP pcimPolicy
     STRUCTURAL
```

```
     MUST ( pcelsPriority )
     MAY ( pcelsPolicySetName
         $ pcelsPolicySetDN )
   )

objectclass ( 1.3.6.1.1.9.1.3
     NAME 'pcelsGroup'
     DESC 'Base class for representing a policy group'
     SUP pcelsPolicySet
     ABSTRACT
     MAY ( pcimGroupName )
   )

objectclass ( 1.3.6.1.1.9.1.4
     NAME 'pcelsGroupAuxClass'
     DESC 'Auxiliary class for representing a policy group'
     SUP pcelsGroup
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.5
     NAME 'pcelsGroupInstance'
     DESC 'Structural class for representing a policy group'
     SUP pcelsGroup
     STRUCTURAL
   )


attributetype ( 1.3.6.1.1.9.2.62
     NAME 'pcelsRuleValidityPeriodList'
     DESC 'Unordered set of DNs of pcimRuleValidityAssociation entries'
     EQUALITY distinguishedNameMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
```

```
   )

attributetype ( 1.3.6.1.1.9.2.6
     NAME 'pcelsConditionListType'
     DESC 'Indicates the type of condition aggregation'
     EQUALITY integerMatch
     ORDERING integerOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.7
     NAME 'pcelsConditionList'
     DESC 'Unordered set of DNs of pcelsConditionAssociation entries'
     EQUALITY distinguishedNameMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
   )

attributetype ( 1.3.6.1.1.9.2.8
     NAME 'pcelsActionList'
     DESC 'Unordered set of DNs of pcelsActionAssociation entries'
     EQUALITY distinguishedNameMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
   )

attributetype ( 1.3.6.1.1.9.2.9
     NAME 'pcelsSequencedActions'
     DESC 'Indicates the importance of action sequencing'
     EQUALITY integerMatch
     ORDERING integerOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
     SINGLE-VALUE
   )
```

```
attributetype ( 1.3.6.1.1.9.2.10
     NAME 'pcelsExecutionStrategy'
     DESC 'Indicates the action execution strategy'
     EQUALITY integerMatch
     ORDERING integerOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
     SINGLE-VALUE
   )

objectclass ( 1.3.6.1.1.9.1.6
     NAME 'pcelsRule'
     DESC 'Base class for representing a policy rule'
     SUP pcelsPolicySet
     ABSTRACT
     MAY ( pcimRuleName
         $ pcimRuleEnabled
         $ pcimRuleUsage
         $ pcimRuleMandatory
         $ pcelsRuleValidityPeriodList
         $ pcelsConditionListType
         $ pcelsConditionList
         $ pcelsActionList
         $ pcelsSequencedActions
         $ pcelsExecutionStrategy )
   )

objectclass ( 1.3.6.1.1.9.1.8
     NAME 'pcelsRuleInstance'
     DESC 'Structural class for representing a policy rule'
     SUP pcelsRule
     STRUCTURAL
   )
```

```
objectclass ( 1.3.6.1.1.9.1.7
     NAME 'pcelsRuleAuxClass'
     DESC 'Auxiliary class for representing a policy rule'
     SUP pcelsRule
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.9
     NAME 'pcelsConditionAssociation'
     DESC 'Associates a policy conditions to an aggregating entry'
     SUP pcimRuleConditionAssociation
     STRUCTURAL
   )

objectclass ( 1.3.6.1.1.9.1.10
     NAME 'pcelsActionAssociation'
     DESC 'Associates a policy conditions to an aggregating entry'
     SUP pcimRuleActionAssociation
     STRUCTURAL
   )

attributetype ( 1.3.6.1.1.9.2.11
     NAME 'pcelsVariableDN'
     DESC 'DN of a pcelsVariable entry'
     EQUALITY distinguishedNameMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.12
     NAME 'pcelsValueDN'
     DESC 'DN of a pcelsValueAuxClass entry'
```

```
      EQUALITY distinguishedNameMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
      SINGLE-VALUE
   )

objectclass ( 1.3.6.1.1.9.1.11
      NAME 'pcelsSimpleConditionAuxClass'
      DESC 'Value matching condition for a policy variable'
      SUP pcimConditionAuxClass
      AUXILIARY
      MAY ( pcelsVariableDN
          $ pcelsValueDN )
   )

objectclass ( 1.3.6.1.1.9.1.12
      NAME 'pcelsCompoundConditionAuxClass'
      DESC 'Boolean combination of simpler conditions'
      SUP pcimConditionAuxClass
      AUXILIARY
      MAY ( pcelsConditionListType
          $ pcelsConditionList )
   )

attributetype ( 1.3.6.1.1.9.2.13
      NAME 'pcelsIsMirrored'
      DESC 'Indicates whether the mirrored traffic matches'
      EQUALITY booleanMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
      SINGLE-VALUE
   )

objectclass ( 1.3.6.1.1.9.1.13
      NAME 'pcelsCompoundFilterConditionAuxClass'
```

```
     DESC 'A compound condition with mirroring capabilities'
     SUP pcelsCompoundConditionAuxClass
     AUXILIARY
     MAY ( pcelsIsMirrored )
   )

objectclass ( 1.3.6.1.1.9.1.14
     NAME 'pcelsSimpleActionAuxClass'
     DESC 'Value assignment action for a policy variable'
     SUP pcimActionAuxClass
     AUXILIARY
     MAY ( pcelsVariableDN
         $ pcelsValueDN )
   )

objectclass ( 1.3.6.1.1.9.1.15
     NAME 'pcelsCompoundActionAuxClass'
     DESC 'Sequence of actions with specific execution strategy'
     SUP pcimActionAuxClass
     AUXILIARY
     MAY ( pcelsActionList
         $ pcelsSequencedActions
         $ pcelsExecutionStrategy )
   )

attributetype ( 1.3.6.1.1.9.2.14
     NAME 'pcelsVariableName'
     DESC 'The user-friendly name of a variable.'
     EQUALITY caseIgnoreMatch
     ORDERING caseIgnoreOrderingMatch
     SUBSTR caseIgnoreSubstringsMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
     SINGLE-VALUE
```

```
   )

attributetype ( 1.3.6.1.1.9.2.15
     NAME 'pcelsExpectedValueList'
     DESC 'Unordered set of DNs of pcelsValueAuxClass entries
           representing expected values for a policy variable'
     EQUALITY distinguishedNameMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
   )

objectclass ( 1.3.6.1.1.9.1.16
     NAME 'pcelsVariable'
     DESC 'Base class for representing a policy variable'
     SUP top
     ABSTRACT
     MAY ( pcelsVariableName
         $ pcelsExpectedValueList )
   )

attributetype ( 1.3.6.1.1.9.2.16
     NAME 'pcelsVariableModelClass'
     DESC 'Identifies a CIM class'
     EQUALITY caseIgnoreMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.17
     NAME 'pcelsVariableModelProperty'
     DESC 'Identifies the property of a CIM class.'
     EQUALITY caseIgnoreMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
     SINGLE-VALUE
```

```
   )

objectclass ( 1.3.6.1.1.9.1.17
     NAME 'pcelsExplicitVariableAuxClass'
     DESC 'Explicitly defined policy variable'
     SUP pcelsVariable
     AUXILIARY
     MUST ( pcelsVariableModelClass
          $ pcelsVariableModelProperty )
   )

attributetype ( 1.3.6.1.1.9.2.18
     NAME 'pcelsExpectedValueTypes'
     DESC 'Identifies subclasses of pcelsValueAuxClass by name'
     EQUALITY caseIgnoreMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   )

objectclass ( 1.3.6.1.1.9.1.18
     NAME 'pcelsImplicitVariableAuxClass'
     DESC 'Implicitly defined policy variable'
     SUP pcelsVariable
     AUXILIARY
     MAY ( pcelsExpectedValueTypes )
   )

objectclass ( 1.3.6.1.1.9.1.19
     NAME 'pcelsSourceIPv4VariableAuxClass'
     DESC 'Source IP v4 address'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )
```

```
objectclass ( 1.3.6.1.1.9.1.20
     NAME 'pcelsSourceIPv6VariableAuxClass'
     DESC 'Source IP v6 address'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.21
     NAME 'pcelsDestinationIPv4VariableAuxClass'
     DESC 'Destination IP v4 address'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.22
     NAME 'pcelsDestinationIPv6VariableAuxClass'
     DESC 'Destination IP v6 address'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.23
     NAME 'pcelsSourcePortVariableAuxClass'
     DESC 'Source port'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.24
     NAME 'pcelsDestinationPortVariableAuxClass'
     DESC 'Destination port'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
```

```
   )

objectclass ( 1.3.6.1.1.9.1.25
     NAME 'pcelsIPProtocolVariableAuxClass'
     DESC 'IP protocol number'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.26
     NAME 'pcelsIPVersionVariableAuxClass'
     DESC 'IP version number'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.27
     NAME 'pcelsIPToSVariableAuxClass'
     DESC 'IP ToS octet'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.28
     NAME 'pcelsDSCPVariableAuxClass'
     DESC 'DiffServ code point'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.29
     NAME 'pcelsFlowIdVariableAuxClass'
     DESC 'Flow Identifier'
```

```
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.30
     NAME 'pcelsSourceMACVariableAuxClass'
     DESC 'Source MAC address'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.31
     NAME 'pcelsDestinationMACVariableAuxClass'
     DESC 'Destination MAC address'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.32
     NAME 'pcelsVLANVariableAuxClass'
     DESC 'VLAN'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.33
     NAME 'pcelsCoSVariableAuxClass'
     DESC 'Class of service'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.34
```

```
     NAME 'pcelsEthertypeVariableAuxClass'
     DESC 'Ethertype'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.35
     NAME 'pcelsSourceSAPVariableAuxClass'
     DESC 'Source SAP'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.36
     NAME 'pcelsDestinationSAPVariableAuxClass'
     DESC 'Destination SAP'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.37
     NAME 'pcelsSNAPOUIVariableAuxClass'
     DESC 'SNAP OUI'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.38
     NAME 'pcelsSNAPTypeVariableAuxClass'
     DESC 'SNAP type'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )
```

```
objectclass ( 1.3.6.1.1.9.1.39
     NAME 'pcelsFlowDirectionVariableAuxClass'
     DESC 'Flow direction'
     SUP pcelsImplicitVariableAuxClass
     AUXILIARY
   )

attributetype ( 1.3.6.1.1.9.2.19
     NAME 'pcelsValueName'
     DESC 'The user-friendly name of a value'
     EQUALITY caseIgnoreMatch
     ORDERING caseIgnoreOrderingMatch
     SUBSTR caseIgnoreSubstringsMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
     SINGLE-VALUE
   )

objectclass ( 1.3.6.1.1.9.1.40
     NAME 'pcelsValueAuxClass'
     DESC 'Base class for representing a policy value'
     SUP top
     AUXILIARY
     MAY ( pcelsValueName )
   )

attributetype ( 1.3.6.1.1.9.2.20
     NAME 'pcelsIPv4AddrList'
     DESC 'Unordered set of IPv4 addresses, IPv4 address ranges or
           hosts'
     EQUALITY caseIgnoreMatch
     ORDERING caseIgnoreOrderingMatch
     SUBSTR caseIgnoreSubstringsMatch
```

```
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   )

objectclass ( 1.3.6.1.1.9.1.41
      NAME 'pcelsIPv4AddrValueAuxClass'
      DESC 'Provides IPv4 addresses'
      SUP pcelsValueAuxClass
      AUXILIARY
      MUST ( pcelsIPv4AddrList )
   )

attributetype ( 1.3.6.1.1.9.2.21
      NAME 'pcelsIPv6AddrList'
      DESC 'Unordered set of IPv6 addresses, IPv6 address ranges or
            hosts'
      EQUALITY caseIgnoreMatch
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   )

objectclass ( 1.3.6.1.1.9.1.42
      NAME 'pcelsIPv6AddrValueAuxClass'
      DESC 'Provides IPv6 addresses'
      SUP pcelsValueAuxClass
      AUXILIARY
      MUST ( pcelsIPv6AddrList )
   )

attributetype ( 1.3.6.1.1.9.2.22
      NAME 'pcelsMACAddrList'
      DESC 'Unordered set of MAC addresses or MAC address ranges'
      EQUALITY caseIgnoreMatch
```

```
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   )

objectclass ( 1.3.6.1.1.9.1.43
      NAME 'pcelsMACAddrValueAuxClass'
      DESC 'Provides MAC addresses'
      SUP pcelsValueAuxClass
      AUXILIARY
      MUST ( pcelsMACAddrList )
   )

attributetype ( 1.3.6.1.1.9.2.23
      NAME 'pcelsStringList'
      DESC 'Unordered set of strings with wildcards'
      EQUALITY caseIgnoreMatch
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   )

objectclass ( 1.3.6.1.1.9.1.44
      NAME 'pcelsStringValueAuxClass'
      DESC 'Provides string values'
      SUP pcelsValueAuxClass
      AUXILIARY
      MUST ( pcelsStringList )
   )

attributetype ( 1.3.6.1.1.9.2.24
      NAME 'pcelsBitStringList'
      DESC 'Unordered set of bit strings or bit string ranges'
```

```
      EQUALITY caseIgnoreMatch
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   )

objectclass ( 1.3.6.1.1.9.1.45
      NAME 'pcelsBitStringValueAuxClass'
      DESC 'Provides bit strings'
      SUP pcelsValueAuxClass
      AUXILIARY
      MUST ( pcelsBitStringList )
   )

attributetype ( 1.3.6.1.1.9.2.25
      NAME 'pcelsIntegerList'
      DESC 'Unordered set of integers or integer ranges'
      EQUALITY caseIgnoreMatch
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
   )

objectclass ( 1.3.6.1.1.9.1.46
      NAME 'pcelsIntegerValueAuxClass'
      DESC 'Provides integer values'
      SUP pcelsValueAuxClass
      AUXILIARY
      MUST ( pcelsIntegerList )
   )

attributetype ( 1.3.6.1.1.9.2.26
      NAME 'pcelsBoolean'
```

```
      DESC 'Boolean value'
      EQUALITY booleanMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
      SINGLE-VALUE
   )

objectclass ( 1.3.6.1.1.9.1.47
      NAME 'pcelsBooleanValueAuxClass'
      DESC 'Provides a boolean value.'
      SUP pcelsValueAuxClass
      AUXILIARY
      MUST ( pcelsBoolean )
   )

attributetype ( 1.3.6.1.1.9.2.27
      NAME 'pcelsReusableContainerName'
      DESC 'User-friendly name of a reusable policy container'
      EQUALITY caseIgnoreMatch
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
      SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.28
      NAME 'pcelsReusableContainerList'
      DESC 'Unordered set of DNs of pcelsReusableContainer entries'
      EQUALITY distinguishedNameMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
   )

objectclass ( 1.3.6.1.1.9.1.48
      NAME 'pcelsReusableContainer'
```

```
     DESC 'Container for reusable policy information'
     SUP pcimRepository
     ABSTRACT
     MAY ( pcelsReusableContainerName
   $ pcelsReusableContainerList )
   )

objectclass ( 1.3.6.1.1.9.1.49
     NAME 'pcelsReusableContainerAuxClass'
     DESC 'Container for reusable policy information'
     SUP pcelsReusableContainer
     AUXILIARY
   )

objectclass ( 1.3.6.1.1.9.1.50
     NAME 'pcelsReusableContainerInstance'
     DESC 'Container for reusable policy information'
     SUP pcelsReusableContainer
     STRUCTURAL
   )

attributetype ( 1.3.6.1.1.9.2.29
     NAME 'pcelsRole'
     DESC 'String representing a role.'
     EQUALITY caseIgnoreMatch
     ORDERING caseIgnoreOrderingMatch
     SUBSTR caseIgnoreSubstringsMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.30
     NAME 'pcelsRoleCollectionName'
```

```
      DESC 'User-friendly name of a role collection'
      EQUALITY caseIgnoreMatch
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
      SINGLE-VALUE
    )

attributetype ( 1.3.6.1.1.9.2.31
      NAME 'pcelsElementList'
      DESC 'Unordered set of managed elements'
      EQUALITY distinguishedNameMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
    )

objectclass ( 1.3.6.1.1.9.1.51
      NAME 'pcelsRoleCollection'
      DESC 'Collection of managed elements that share a common role'
      SUP pcimPolicy
      STRUCTURAL
      MUST ( pcelsRole )
      MAY ( pcelsRoleCollectionName
    $ pcelsElementList )
    )

attributetype ( 1.3.6.1.1.9.2.32
      NAME 'pcelsFilterName'
      DESC 'User-friendly name of a filter entry'
      EQUALITY caseIgnoreMatch
      ORDERING caseIgnoreOrderingMatch
      SUBSTR caseIgnoreSubstringsMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
      SINGLE-VALUE
```

```
   )

attributetype ( 1.3.6.1.1.9.2.33
     NAME 'pcelsFilterIsNegated'
     DESC 'Indicates whether the filter is negated'
     EQUALITY booleanMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
     SINGLE-VALUE
   )

objectclass ( 1.3.6.1.1.9.1.52
     NAME 'pcelsFilterEntryBase'
     DESC 'Base class for message or packet filters'
     SUP pcimPolicy
     ABSTRACT
     MAY ( pcelsFilterName
         $ pcelsFilterIsNegated )
   )

attributetype ( 1.3.6.1.1.9.2.34
     NAME 'pcelsIPHdrVersion'
     DESC 'IP version'
     EQUALITY integerMatch
     ORDERING integerOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.35
     NAME 'pcelsIPHdrSourceAddress'
     DESC 'Source IP address'
     EQUALITY octetStringMatch
     ORDERING octetStringOrderingMatch
```

```
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
      SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.36
      NAME 'pcelsIPHdrSourceAddressEndOfRange'
      DESC 'End of a range of source IP addresses'
      EQUALITY octetStringMatch
      ORDERING octetStringOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
      SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.37
      NAME 'pcelsIPHdrSourceMask'
      DESC 'Mask to be used in comparing the source IP address'
      EQUALITY octetStringMatch
      ORDERING octetStringOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
      SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.38
      NAME 'pcelsIPHdrDestAddress'
      DESC 'Destination IP address'
      EQUALITY octetStringMatch
      ORDERING octetStringOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
      SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.39
      NAME 'pcelsIPHdrDestAddressEndOfRange'
```

```
      DESC 'End of a range of destination IP addresses'
      EQUALITY octetStringMatch
      ORDERING octetStringOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
      SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.40
      NAME 'pcelsIPHdrDestMask'
      DESC 'Mask to be used in comparing the destination IP address'
      EQUALITY octetStringMatch
      ORDERING octetStringOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
      SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.41
      NAME 'pcelsIPHdrProtocolID'
      DESC 'IP protocol type'
      EQUALITY integerMatch
      ORDERING integerOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
      SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.42
      NAME 'pcelsIPHdrSourcePortStart'
      DESC 'Lower end of a range of UDP or TCP source ports'
      EQUALITY integerMatch
      ORDERING integerOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
      SINGLE-VALUE
   )
```

```
attributetype ( 1.3.6.1.1.9.2.43
     NAME 'pcelsIPHdrSourcePortEnd'
     DESC 'Upper end of a range of UDP or TCP source ports'
     EQUALITY integerMatch
     ORDERING integerOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.44
     NAME 'pcelsIPHdrDestPortStart'
     DESC 'Lower end of a range of UDP or TCP destination ports'
     EQUALITY integerMatch
     ORDERING integerOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.45
     NAME 'pcelsIPHdrDestPortEnd'
     DESC 'Upper end of a range of UDP or TCP destination ports'
     EQUALITY integerMatch
     ORDERING integerOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.46
     NAME 'pcelsIPHdrDSCPList'
     DESC 'DSCP values'
     EQUALITY integerMatch
     ORDERING integerOrderingMatch
```

```
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
   )

attributetype ( 1.3.6.1.1.9.2.47
      NAME 'pcelsIPHdrFlowLabel'
      DESC 'IP flow label'
      EQUALITY octetStringMatch
      ORDERING octetStringOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
      SINGLE-VALUE
   )

objectclass ( 1.3.6.1.1.9.1.53
      NAME 'pcelsIPHeadersFilter'
      DESC 'IP header filter'
      SUP pcelsFilterEntryBase
      STRUCTURAL
      MAY ( pcelsIPHdrVersion
          $ pcelsIPHdrSourceAddress
          $ pcelsIPHdrSourceAddressEndOfRange
          $ pcelsIPHdrSourceMask
          $ pcelsIPHdrDestAddress
          $ pcelsIPHdrDestAddressEndOfRange
          $ pcelsIPHdrDestMask
          $ pcelsIPHdrProtocolID
          $ pcelsIPHdrSourcePortStart
          $ pcelsIPHdrSourcePortEnd
          $ pcelsIPHdrDestPortStart
          $ pcelsIPHdrDestPortEnd
          $ pcelsIPHdrDSCPList
          $ pcelsIPHdrFlowLabel )
   )
```

```
attributetype ( 1.3.6.1.1.9.2.48
     NAME 'pcels8021HdrSourceMACAddress'
     DESC 'Source MAC address'
     EQUALITY octetStringMatch
     ORDERING octetStringOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.49
     NAME 'pcels8021HdrSourceMACMask'
     DESC 'Source MAC address mask'
     EQUALITY octetStringMatch
     ORDERING octetStringOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.50
     NAME 'pcels8021HdrDestMACAddress'
     DESC 'Destination MAC address'
     EQUALITY octetStringMatch
     ORDERING octetStringOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.51
     NAME 'pcels8021HdrDestMACMask'
     DESC 'Destination MAC address mask'
     EQUALITY octetStringMatch
     ORDERING octetStringOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
```

```
      SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.52
      NAME 'pcels8021HdrProtocolID'
      DESC 'Ethernet protocol ID'
      EQUALITY integerMatch
      ORDERING integerOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
   )

attributetype ( 1.3.6.1.1.9.2.53
      NAME 'pcels8021HdrPriority'
      DESC '802.1Q priority'
      EQUALITY integerMatch
      ORDERING integerOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
   )

attributetype ( 1.3.6.1.1.9.2.54
      NAME 'pcels8021HdrVLANID'
      DESC '802.1Q VLAN ID'
      EQUALITY integerMatch
      ORDERING integerOrderingMatch
      SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
   )

objectclass ( 1.3.6.1.1.9.1.54
      NAME 'pcels8021Filter'
      DESC '802.1 header filter'
      SUP pcelsFilterEntryBase
      STRUCTURAL
      MAY ( pcels8021HdrSourceMACAddress
```

```
        $ pcels8021HdrSourceMACMask
        $ pcels8021HdrDestMACAddress
        $ pcels8021HdrDestMACMask
        $ pcels8021HdrProtocolID
        $ pcels8021HdrPriority
        $ pcels8021HdrVLANID )
   )

attributetype ( 1.3.6.1.1.9.2.55
     NAME 'pcelsFilterListName'
     DESC 'User-friendly name of a FilterList'
     EQUALITY caseIgnoreMatch
     ORDERING caseIgnoreOrderingMatch
     SUBSTR caseIgnoreSubstringsMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.56
     NAME 'pcelsFilterDirection'
     DESC 'Direction to which this filter is applied'
     EQUALITY integerMatch
     ORDERING integerOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
     SINGLE-VALUE
   )

attributetype ( 1.3.6.1.1.9.2.57
     NAME 'pcelsFilterEntryList'
     DESC 'Unordered set of DNs of pcelsFilterEntryBase entries'
     EQUALITY distinguishedNameMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
   )
```

```
objectclass ( 1.3.6.1.1.9.1.55
     NAME 'pcelsFilterListAuxClass'
     DESC 'Collection of pcelsFilterEntryBase filters'
     SUP pcimConditionAuxClass
     AUXILIARY
     MAY ( pcelsFilterListName
         $ pcelsFilterDirection
         $ pcelsFilterEntryList )
   )

attributetype ( 1.3.6.1.1.9.2.58
     NAME 'pcelsVendorVariableData'
     DESC 'Mechanism for representing variables that have not
           been specifically modeled'
     EQUALITY octetStringMatch
     ORDERING octetStringOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
   )

attributetype ( 1.3.6.1.1.9.2.59
     NAME 'pcelsVendorVariableEncoding'
     DESC 'Identifies the format and semantics for policy variables'
     EQUALITY objectIdentifierMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
     SINGLE-VALUE
   )

objectclass ( 1.3.6.1.1.9.1.56
     NAME 'pcelsVendorVariableAuxClass'
     DESC 'Defines registered means to describe a policy variable'
     SUP pcelsVariable
     AUXILIARY
```

```
     MAY ( pcelsVendorVariableData $
           pcelsVendorVariableEncoding )
   )

attributetype ( 1.3.6.1.1.9.2.60
     NAME 'pcelsVendorValueData'
     DESC 'Mechanism for representing values that have not been
           specifically modeled'
     EQUALITY octetStringMatch
     ORDERING octetStringOrderingMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
   )

attributetype ( 1.3.6.1.1.9.2.61
     NAME 'pcelsVendorValueEncoding'
     DESC 'Identifies the format and semantics for policy values'
     EQUALITY objectIdentifierMatch
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
     SINGLE-VALUE
   )

objectclass ( 1.3.6.1.1.9.1.57
     NAME 'pcelsVendorValueAuxClass'
     DESC 'Defines registered means to describe a policy value'
     SUP pcelsValueAuxClass
     AUXILIARY
     MAY ( pcelsVendorValueData $
           pcelsVendorValueEncoding )
   )
```

7.3 Configuração Bind (DNS VNET.LAB)

7.3.1 DNS Reversal zones

```
$TTL 86400
@ INSOA  serverldap.vnet.lab. root.vnet.lab. (
    100 ; Serial
    3H ; refresh
    15M ; retry
    1W ; expiry
    1D ) ; minimum

  IN   NS   serverldap.vnet.lab.

1 IN   PTR routerdc
1 IN   PTR PEP1
50   IN   PTR PR
50   IN   PTR serverldap
51   IN   PTR servermain
51   IN   PTR PMT
52   IN   PTR serverapp1
53   IN   PTR serverapp2
54   IN   PTR dns2
54   IN   PTR solaris10
54   IN   PTR serverapp3
54   IN   PTR mail
```

7.3.2 DNS zones

```
$TTL 86400
@   IN   SOA serverldap.vnet.lab. root.vnet.lab. (
        100 ;Serial
        3H ; refresh
        15M ; retry
        1W ; expiry
        1D ) ; minimum

    INNS   serverldap.vnet.lab.
    IN   MX 10  vnet.lab.

localhost   IN   A 127.0.0.1
vnet.lab  IN   A 10.10.1.50
routerdc  IN   A 10.10.1.1
PEP1     IN   A 10.10.1.1
serverldap  IN   A 10.10.1.50
PR     IN   A 10.10.1.50
servermain  IN   A 10.10.1.51
PMT    IN   A 10.10.1.51
dns    IN   A 10.10.1.50
serverapp1  IN   A 10.10.1.52
serverapp2  IN   A 10.10.1.53
dns2     IN   A 10.10.1.54
solaris10   IN   A 10.10.1.54
mail     IN   A 10.10.1.54
serverapp3  IN   A 10.10.1.54
routernet INA 10.10.4.1
routerst1 INA 10.10.2.1
routerst2 INA 10.10.3.1
PEP2   INA 10.10.2.1
```

```
PEP3   INA 10.10.3.1
PEP4   INA 10.10.4.1
```

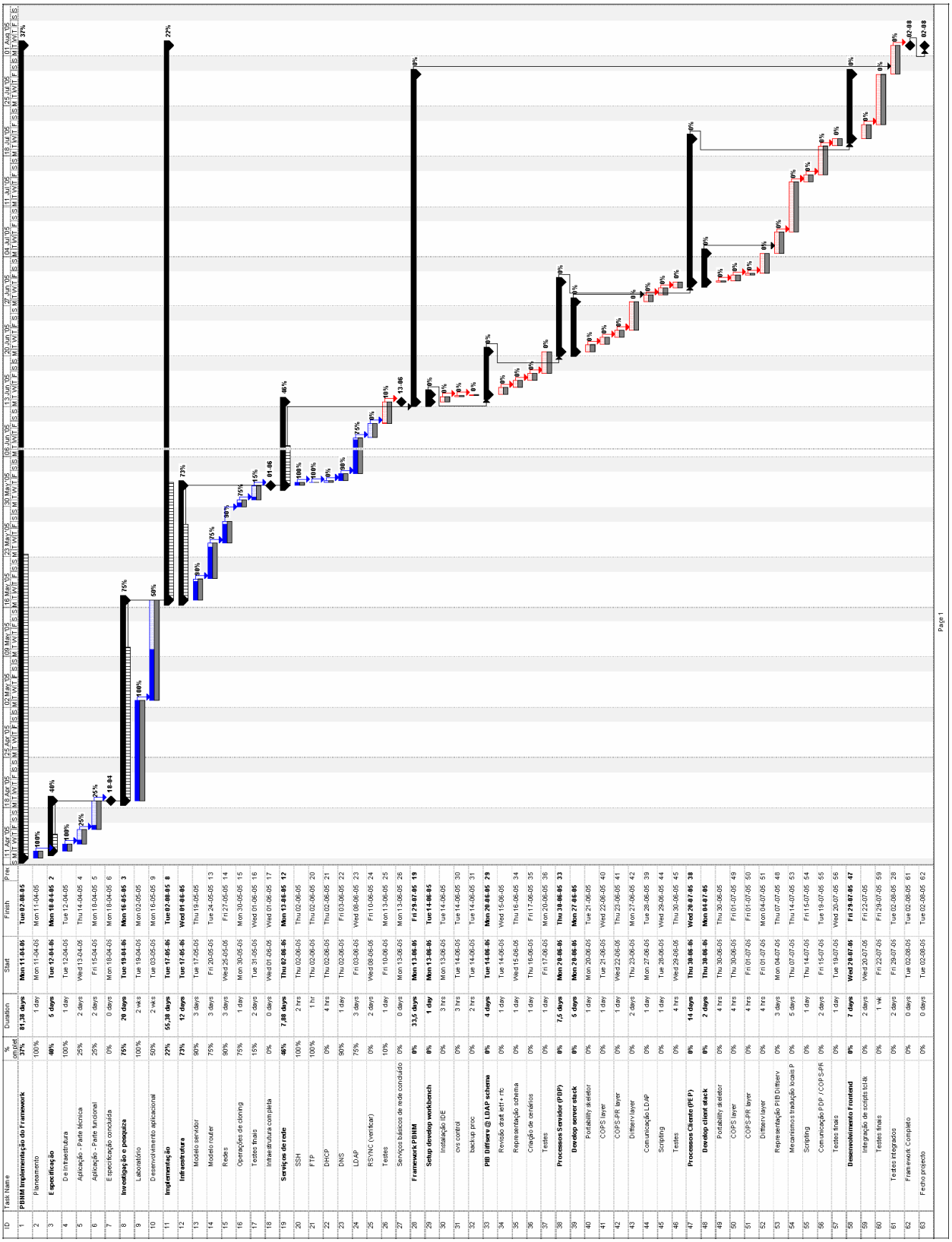
7.3.3 Configuração Bind

```
options {
  directory "/var/named";
  forwarders { 192.168.100.2; 10.240.128.46; };
  version "isc bind 9.2.3";
  // minimal-responses yes;
  listen-on { 10.10.1.50; };
  allow-transfer { 10.10.1.54; };
  allow-recursion {127.0.0.1; localhost; 10.10.0.0/16;};

  /*
   * If there is a firewall between you and nameservers you want
   * to talk to, you might need to uncomment the query-source
   * directive below.  Previous versions of BIND always asked
   * questions using port 53, but BIND 8.1 uses an unprivileged
   * port by default.
   */
  query-source address * port 53;
};

//
// a caching only nameserver config
//
zone "." IN {
  type hint;
  file "caching-example/named.ca";
};

zone "localhost" IN {
```

```
  type master;
  file "caching-example/localhost.zone";
  allow-update { none; };
};

zone "0.0.127.in-addr.arpa" IN {
  type master;
  file "caching-example/named.local";
  allow-update { none; };
};

zone "0.1.10.10.in-addr.arpa" IN {
type master;
file "vnet-lab/vnet.rev";
allow-transfer { 10.10.1.54; };
allow-query { any; };
};

zone "vnet.lab" IN {
type master;
file "vnet-lab/vnet.host";
allow-transfer { 10.10.1.54 ; };
allow-query { any; };
};
```

7.4 Proposta de Planeamento

7.4.1 Gráfico de Gantt



7.4.2 Relação de Tarefas

Tasks with notes (GAT) as of Tue 07-06-05
plano implementação v1

ID	Code	Task Name	% Complete	Duration	Start	Finish	Predec
2	A-1	Planeamento	100%	1 day	Mon 11-04-05	Mon 11-04-05	
4	A-2.1	De Infraestrutura	100%	1 day	Tue 12-04-05	Tue 12-04-05	
5	A-2.2	Aplicação - Parte técnica	25%	2 days	Wed 13-04-05	Thu 14-04-05	4
6	A-2.3	Aplicação - Parte funcional	25%	2 days	Fri 15-04-05	Mon 18-04-05	5
7	A-2.4	Especificação concluída	0%	0 days	Mon 18-04-05	Mon 18-04-05	6
9	A-3.1	Laboratório	100%	2 wks	Tue 19-04-05	Mon 02-05-05	
10	A-3.2	Desenvolvimento aplicacional	50%	2 wks	Tue 03-05-05	Mon 16-05-05	9
13	A-4.1.1	Modelo servidor	90%	3 days	Tue 17-05-05	Thu 19-05-05	
14	A-4.1.2	Modelo router	75%	3 days	Fri 20-05-05	Tue 24-05-05	13
15	A-4.1.3	Redes	90%	3 days	Wed 25-05-05	Fri 27-05-05	14
16	A-4.1.4	Operações de cloning	75%	1 day	Mon 30-05-05	Mon 30-05-05	15
17	A-4.1.5	Testes finais	15%	2 days	Tue 31-05-05	Wed 01-06-05	16
18	A-4.1.6	Infraestrutura completa	0%	0 days	Wed 01-06-05	Wed 01-06-05	17
20	A-4.2.1	SSH	100%	2 hrs	Thu 02-06-05	Thu 02-06-05	
21	A-4.2.2	FTP	100%	1 hr	Thu 02-06-05	Thu 02-06-05	20
	proftpd						
22	A-4.2.3	DHCP	0%	4 hrs	Thu 02-06-05	Thu 02-06-05	21
23	A-4.2.4	DNS	90%	1 day	Thu 02-06-05	Fri 03-06-05	22
	bind 9 - processo named. - config concluída.						
24	A-4.2.5	LDAP	75%	3 days	Fri 03-06-05	Wed 08-06-05	23
	openldap - concluído. falta implementar schemas espceíficos para pib's, etc... autenticação mínima para validar users nos postos clientes.						
25	A-4.2.6	RSYNC (verificar)	0%	2 days	Wed 08-06-05	Fri 10-06-05	24
	serviço para sincronizar várias directorias pelos routers e servers clones trabalhando apenas em 1 só.						
26	A-4.2.7	Testes	10%	1 day	Fri 10-06-05	Mon 13-06-05	25
27	A-4.2.8	Serviços básicos de rede concluídos	0%	0 days	Mon 13-06-05	Mon 13-06-05	26
30	A-4.3.1.1	Instalação IDE	0%	3 hrs	Mon 13-06-05	Tue 14-06-05	
31	A-4.3.1.2	cvs control	0%	3 hrs	Tue 14-06-05	Tue 14-06-05	30
32	A-4.3.1.3	backup proc	0%	2 hrs	Tue 14-06-05	Tue 14-06-05	31
34	A-4.3.2.1	Revisão draft ietf + rfc	0%	1 day	Tue 14-06-05	Wed 15-06-05	
35	A-4.3.2.2	Representação schema	0%	1 day	Wed 15-06-05	Thu 16-06-05	34
36	A-4.3.2.3	Criação de cenários	0%	1 day	Thu 16-06-05	Fri 17-06-05	35
37	A-4.3.2.4	Testes	0%	1 day	Fri 17-06-05	Mon 20-06-05	36
40	A-4.3.3.1.1	Portability skeletor	0%	1 day	Mon 20-06-05	Tue 21-06-05	
41	A-4.3.3.1.2	COPS layer	0%	1 day	Tue 21-06-05	Wed 22-06-05	40
42	A-4.3.3.1.3	COPS-PR layer	0%	1 day	Wed 22-06-05	Thu 23-06-05	41
43	A-4.3.3.1.4	Diffserv layer	0%	2 days	Thu 23-06-05	Mon 27-06-05	42
44	A-4.3.3.2	Comunicação LDAP	0%	1 day	Mon 27-06-05	Tue 28-06-05	39
45	A-4.3.3.3	Scripting	0%	1 day	Tue 28-06-05	Wed 29-06-05	44
46	A-4.3.3.4	Testes	0%	4 hrs	Wed 29-06-05	Thu 30-06-05	45
49	A-4.3.4.1.1	Portability skeletor	0%	4 hrs	Thu 30-06-05	Thu 30-06-05	
50	A-4.3.4.1.2	COPS layer	0%	4 hrs	Thu 30-06-05	Fri 01-07-05	49
51	A-4.3.4.1.3	COPS-PR layer	0%	4 hrs	Fri 01-07-05	Fri 01-07-05	50
52	A-4.3.4.1.4	Diffserv layer	0%	4 hrs	Fri 01-07-05	Mon 04-07-05	51
53	A-4.3.4.2	Representação PIB Diffserv	0%	3 days	Mon 04-07-05	Thu 07-07-05	48
54	A-4.3.4.3	Mecanismos tradução locais PEP	0%	5 days	Thu 07-07-05	Thu 14-07-05	53
55	A-4.3.4.4	Scripting	0%	1 day	Thu 14-07-05	Fri 15-07-05	54
56	A-4.3.4.5	Comunicação PDP / COPS-PR	0%	2 days	Fri 15-07-05	Tue 19-07-05	55
57	A-4.3.4.6	Testes finais	0%	1 day	Tue 19-07-05	Wed 20-07-05	56
59	A-4.3.5.1	Integração de scripts tcl-tk	0%	2 days	Wed 20-07-05	Fri 22-07-05	
60	A-4.3.5.2	Testes finais	0%	1 wk	Fri 22-07-05	Fri 29-07-05	59
61	A-4.4	Testes integrados	0%	2 days	Fri 29-07-05	Tue 02-08-05	28
62	A-4.5	Framework Completo	0%	0 days	Tue 02-08-05	Tue 02-08-05	61
63	A-5	Fecho projecto	0%	0 days	Tue 02-08-05	Tue 02-08-05	62