INSTITUTO UNIVERSITÁRIO DA MAIA

Estudo e implementação de uma infraestrutura de FCAPS na rede da Universidade do Porto

JOÃO RICARDO GEITOEIRA LOPES

Licenciado em Informática – Especialização em Redes de Nova Geração
no Instituto Universitário da Maia

Relatório de Projeto submetido ao Departamento de Ciências da Comunicação e Tecnologias da Informação do Instituto Universitário da Maia para satisfação parcial dos requisitos do grau de Mestre em Tecnologias da Comunicação, Informação e Multimédia – Ramo Telecomunicações

Relatório de Projeto realizado sob a supervisão do Mestre Mário Paulo Monteiro Serrão, do Departamento de Ciências da Comunicação e Tecnologias da Informação do Instituto Universitário da Maia, e do Engenheiro Fernando Marques Correia, Diretor do Serviço de Infraestruturas Tecnológicas da Universidade do Porto

Maia, outubro de 2020

Resumo

A dimensão dos recursos aplicacionais suportados pelas redes de comunicação desencadeia dificuldades de resposta que se traduzem no controlo, administração e monitorização, de modo a proporcionar elevados níveis de disponibilidade e qualidade dos serviços prestados.

Assim, nos dias de hoje, torna-se cada vez mais importante reduzir os desafios operacionais, os quais, do ponto de vista da gestão e manutenção de uma rede de dados, compreendem a diminuição do tempo de inatividade desta, pois o contrário resultará em perdas de produtividade. Nesta circunstância, a gestão de rede proficiente e de alto desempenho, caracteriza-se num fator diferenciador crítico para todos os utilizadores de rede.

A abrangência, a diversidade de sistemas e a complexidade das redes atuais, exige o planeamento de uma rede de gestão informática, considerando um conjunto de mecanismos, protocolos e ferramentas de monitorização para controlo dos recursos de comunicação. Tendo em conta estes pressupostos, considerou-se a utilização de uma abordem pragmática, comumente utilizada e denominada por FCAPS. Este modelo foi utilizado ao longo deste relatório para analisar, descrever e organizar as aplicações, no que respeita ao cumprimento das funcionalidades preconizadas pelas cinco áreas funcionais que o compõe.

Como princípio basilar deste projeto, enumera-se o estudo realizado à gestão de rede da Universidade do Porto (U. Porto), especificamente à infraestrutura protocolar e tecnológica que a apoia e dá suporte às tarefas que descrevem a gestão operacional diária.

Durante a abordagem prática, numa primeira interação, implementou-se um conjunto de ferramentas, procedendo-se posteriormente ao estudo comparativo para avaliá-las. Numa segunda fase, com os resultados apurados, desenvolveu-se uma infraestrutura de monitorização centralizada e ajustada às necessidades críticas da U. Porto. Esta solução foi estruturada em modelos arquitetónicos altamente resilientes (*clusters*) e segmentada em três componentes computacionais, nomeadamente composta por sistemas de armazenamento (Percona-XtraDB-Cluster), de encaminhamento (ProxySQL) e de consulta de dados monitorizados (Zabbix).

Finalmente, são sugeridas um conjunto de ações que, na perspetiva do autor, visam a continuação do processo de melhoria iniciado no âmbito do presente projeto.

Palavras-chave: U. Porto, rede de dados, gestão de rede centralizada, FCAPS e resiliência.

Abstract

The magnitude of the application resources supported by communications networks triggers response difficulties that translates into control, administration, and monitoring, in order to provide high availability and quality of the services that are provided.

Thus, nowadays, it becomes increasingly important to reduce operational challenges, which, from the point of view of the management and maintenance of data network, understands the reduction of its downtime, as the opposite will result in productivity losses. In this circumstance, proficient and high-performance network management is characterized by a critical differentiating factor for all network users.

The scope, the diversity of systems and the complexity of current networks, requires the planning of a computer management network, considering a set of mechanisms, protocols and monitoring tools to control communication resources. Having these assumptions, the use of a pragmatic approach, commonly used and called FCAPS, was considered. This model was used throughout this report to analyze, describe and organize applications, regard to compliance with the functionalities recommended by the five functional areas that comprise it.

As a basic principle of this project, the study carried out on the network management of the University of Porto (U. Porto) is listed, specifically the protocol and technological infrastructure that supports the tasks that describe the daily operational management.

During the practical approach, in a first interaction, a set of tools was implemented, and then a comparative study was carried out to evaluate them. In a second phase, with the results obtained, a centralized monitoring infrastructure was developed and adjusted to the critical needs of the U. Porto. This solution was structured in highly resilient architectural models (clusters) and segmented into three computational components, namely composed of storage systems (Percona-XtraDB-Cluster), routing (ProxySQL) and monitored data query (Zabbix).

Finally, a set of actions are suggested that, in the author's perspective, aim at the continuation of the improvement process initiated within the scope of this project.

Keywords: U. Porto, data network, centralized network management, FCAPS and resilience.

Agradecimentos

Em primeiro lugar, quero agradecer ao orientador deste projeto, o professor Mário Serrão, pelo apoio contínuo ao longo da licenciatura e do mestrado. Nunca irei esquecer as suas aulas e a atenção que me despertavam. Se hoje tenho o encanto pelas "redes" a si o devo.

Manifesto também o meu agradecimento ao coorientador do projeto, o engenheiro Fernando Correia, à Universidade do Porto e ao ISMAI, entre os quais, providenciaram todos os recursos necessários à execução deste relatório.

Destaco ainda um reconhecimento de gratidão especial, ao Telmo Morais e ao Paulo Carvalho da Unidade de Infraestruturas Tecnológicas da U.Porto, pois sem as dicas construtivas e as horas dedicadas do Telmo, bem como, o preciosismo e a fundamentação teórica do Paulo, jamais teria sabido vestir a vossa camisola de "Sysadmin".

Aos meus colegas de curso, particularmente, ao Diogo Pereira e ao Diogo Teixeira, deixo um enorme obrigado, não só pela paciência que tiveram comigo, como também pelo apoio prestado academicamente e profissionalmente.

À Soraia, um enorme pedido de desculpas pela ausência que este projeto causou e um enorme obrigado pelo apoio incondicional e pelas palavras positivas nos piores momentos que passamos recentemente. Não tenho palavras para te agradecer, estás sempre lá.

Por último, destaco com enorme gratidão a minha família, em especial um agradecimento dirigido há minha mãe Isabel e ao meu irmão Diogo. Peço imensa desculpa por ter "desaparecido" durante estes dois anos.

Concluo assim aquela que para mim foi a etapa mais difícil de todo o percurso académico, não podendo deixar de agradecer a todos os que contribuíram direta ou indiretamente neste projeto. Um grande obrigado.

Índice

Lista de figuras

Lista de tabelas

Lista de siglas e abreviaturas

3DES	*Triple Data Encryption Standard*
AAA	*Authentication, Autorisation, Accounting*
AC	*Attatchment circuit*
ACL	*Access Control Lists*
Admin	*Administrador*
AES	*Advanced Encryption Standard*
AMQP	*Advanced Message Queuing Protocol*
AP	*Acess-Point*
APC	*American Power Conversion*
API	*Application Programming Interface*
ARP	*Address Resolution Protocol*
AS	*Autonomous System*
ASA	*Adaptive Security Appliance*
ASN.1	*Abstract Syntax Notation One*
ASR	*Aggregation Services Routers*
BE	*Bundle Ethernet*
BEEP	*Blocks Extensible Exchange Protocol*
BGP	*Border Gateway Protocol*
BSD	*Berkeley Software Distribution*
BVI	*Bridge Virtual Interface*
CAUP	Centro de Astrofísica da Universidade do Porto
CBC	*Cipher Block Chaining*
CCITT	*Consultative Committee for International Telephony and Telegraphy*
CDP	*Cisco Discovery Protocol*
CDUP	Centro de Desporto da Universidade do Porto
CEF	*Common Event Format*
CEMUP	Centro de Materiais da Universidade do Porto
CGI	*Common Gateway Interface*
CIIMAR	Centro Interdisciplinar de Investigação Marinha e Ambiental
CIM	*Common Information Model*

CIPES	Centro de Investigação de Políticas de Ensino Superior
CLI	*Command-Line Interface*
CMAN	*Cluster Manager*
CMDB	*Configuration Management Database*
CMIP	*Common Management Information Protocols*
CMIS	*Common Management Information Service*
CMOT	*Common management information protocol*
Conf	Configuração
COPS	*Common Open Policy Service*
COPS-PR	*COPS Usage for Provisioning*
CORBA	*Common Object Request Broker Architecture*
CPE	*Costumer Provider Edge*
CPU	*Central Process Unit*
CUP	Clínica Universitária de Psicologia
CVS	*Concurrent Versions System*
DB	*Database*
DBA	*Database administrator*
DBMS	*Database Management System*
DCF	*Data Communication Functions*
DES	*Data Encryption Standard*
DEV	*Development servers*
DHCP	*Dynamic Host Configuration Protocol*
DMTF	*Distributed Management Task Force*
DNS	*Domain Name Server*
Dom	*Control Domain*
EBGP	*External Border Gateway Protocol*
EGP	Escola de Gestão do Porto
EGP	*Exterior Gateway Protocol*
ERP	*Enterprise Resource Planning*
etc	*Et cetera*
FADEUP	Faculdade de Desporto da Universidade do Porto
FAUP	Faculdade de Arquitetura da Universidade do Porto
FBAUP	Faculdade de Belas Artes da Universidade do Porto

FCAPS	*Fault-management, Configuration, Accounting, Performance and Security*
FCCN	Fundação para Computação Científica Nacional
FCNAUP	Faculdade de Ciências da Nutrição e Alimentação da Universidade do Porto
FCT	Fundação para a Ciência e a Tecnologia
FCUP	Faculdade de Ciências da Universidade do Porto
FDUP	Faculdade de Direito da Universidade do Porto
FEC	*Forwarding equivalence class*
FEP	Faculdade de Economia do Porto
FEUP	Faculdade de Engenharia da Universidade do Porto
FEX	*Fabric Extenders*
FFUP	Faculdade de Farmácia da Universidade do Porto
FIFO	*First In, First Out*
FIMS	Fundação Instituto Marques da Silva
FLUP	Faculdade de Letras da Universidade do Porto
FMDUP	Faculdade de Medicina Dentária da Universidade do Porto
FMUP	Faculdade de Medicina da Universidade do Porto
FPCEUP	Faculdade de Psicologia e de Ciências da Educação da Universidade do Porto
FTP	*File Transfer Protocol*
FW	*Firewall*
G	Gestão
GB	*Gigabyte*
GCSP	*Group Communication Systems Plugins*
GELF	*Graylog Extended Log Format*
GIOP	*General Inter-ORB Protocol*
GPL	*General Public License*
GRP	*Group Replication Plugins*
GTID	*Global Transaction ID*
GUI	*Graphical User Interface*
HA	*High-Availability*
HG	*Host Group*
HMAC	*Hash-based Message Authentication Code*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transpot Protocol*

IBGP	*Internal Border Gateway Protocol*
IBM	*In-Band Management*
ICBAS	Instituto de Ciências Biomédicas Abel Salazar
ICETA	Instituto de Ciências, Tecnologias e Agroambiente
ICMP	*Internet Control Message Protocol*
ID	*Identification*
IDL	*Interface Definition Language*
IETF	*Internet Engineering Task Force*
IGMP	*Internet Group Management Protocol*
IGP	*Interior Gateway Protocol*
IKE	*Internet Key Exchange*
ILO	*Integrated Lights Out*
IMAP	*Internet Message Access Protocol*
INEGI	Instituto de Ciência e Inovação em Engenharia Mecânica e Engenharia Industrial
INESC	Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência
IP	*Internet Protocol*
IPMI	*Intelligent Platform Management Interface*
IPSec	*IP Security*
IPVS	*IP Virtual Server*
IPX	*Internetwork Packet Exchange*
IRC	*Internet Relay Chat*
ISAKMP	*Internet Security Association and Key Management Protocol*
ISPUP	Instituto de Saúde Pública da Universidade do Porto
ISSO	*International Organization for Standardization*
IT	*Information Technology*
ITU	*International Telecommunication Union*
JMX	*Java Management Extensions*
JSON	*JavaScript Object Notation*
KMS	*Key Management Service*
KVM	*Kernel Virtual Machine*
L2TP	*Layer 2 Tunneling Protocol*
LAG	*Link Aggregation Group*

LAN	*Local Area Network*
LDAP	*Lightweight Directory Access Protocol*
LDP	*Label Distribution Protocol*
LER	*Label Edge Router*
LFIB	*Label Forwarding Information Base*
LIACC	Laboratório de Inteligência Artificial e Ciência de Computadores
LLD	*Low-level Discovery*
LME	*Layer Management Entities*
LPDP	*Local Policy Decision Point*
LSN	*Log Sequence Number*
LSP	*Label Switched Path*
LSR	*Label Switching Router*
Ltd	*Limited*
LVS	*Linux Virtual Server*
LWAPP	*Lightweight Access Point Protocol*
LXC	*Linux Containers*
MAC	*Media Access Control*
MCA	*MariaDB Contributor Agreement*
MD	*Managed Device*
MD5	*Message-Digest algorithm 5*
MF	*Mediation Functions*
MIB	*Management Information Base*
MOF	*Managed Object Format*
MPLS	*Multi Protocol Label Switching*
NAT	*Network Address Translation*
NE	*Network Element*
NEF	*Network Element Functions*
NET	Rede
NETCONF	*Network Configuration Protocol*
NIC	*Network Interface Card*
NMD	*Network Managed Device*
NMS	*Network Management System*
NOC	*Network Operations Center*

NRPE	*Nagios Remote Plugin Executor*
OCA	*Oracle Contributor Agreement*
ODBC	*Open Database Connectivity*
OID	*Object Identifier*
OLAP	*Online Analytical Processing*
OLTP	*Online Transaction Processing*
OOBM	*Out-of-Band Management*
ORB	*Object Request Broker*
OS	*Operative System*
OSF	*Operation System Funcions*
OSI	*Open System Interconnection*
OSPF	*Open Shortest Path First*
OUP	Orfeão Universitário do Porto
P	Percona
PP	*proxy* Percona
P1	Pólo 1
P2	Pólo 2
P3	Pólo 3
PBNM	*Policy-based network management*
PBS	Porto *Business School*
PDP	*Policy Decision Point*
PDU	*Protocol Data Unit*
PEP	*Policy Enforcement Point*
PEs	*Provider Edges*
PGDG	PostgreSQL *Global Development Group*
PHP	*Hypertext Preprocessor*
PIP	*Policy Information Point*
PMC	*Policy Management Console*
PMM	*Percona Monitoring & Management*
POP	*Post Office Protocol*
PXC	Percona XtraDB *cluster*
QAF	*Adaptor Functions*
RADIUS	*Remote Authentication Dial In User Service*

RAM	*Random Access Memory*
RANCID	*Really Awesome New Cisco confIg Differ*
RCTS	Rede, Ciência, Tecnologia e Sociedade
REST	*Representational state transfer*
RFC	*Request for Comments*
RMON	*Remote Network Monitoring*
RPC	*Remote Procedure Call*
RRA	*Round Robin Archives*
RRD	*Round-Robin Database*
RT	*Routers*
S	Servidor
SASUP	Serviços de Ação Social da Universidade do Porto
SDN	*Software Defined Network*
SFP	*Small Form-factor Pluggable*
SGBD	Sistema de Gestão de Base de Dados
SGMP	*Simple Gateway Monitoring Protocol*
SGR	Sistema de Gestão de Redes
SHA	*Secure Hash Algorithms*
SIGARRA	Sistema de Informação para Gestão Agregada dos Recursos e dos Registos Académicos
SMAE	*Systems Management Application Entities*
SMAP	*Systems Management Application Processes*
SMI	*Structure of Management Information*
SMS	*Short Message Service*
SMTP	*Simple Mail Transfer Protocol*
SNMP	*Simple Network Management Protocol*
SOAP	*Simple Object Access Protocol*
SQL	*Structured Query Language*
SRC	Serviços de Redes Centrais e Datacenters
SSH	*Secure Shell*
SSHD	*Solid State Hybrid Disc*
SSID	*Service Set Identifier*
SSL	*Secure Socket Layer*

SST	*State Snapshot Transfer*
STDOUT	*Standard Output*
SVN	*SubVersionN*
SW	*Switch*
TB	*Terabyte*
TCL	*Tool Command Language*
TCP	*Transmission Control Protocol*
TFTP	*Trivial File Transfer Protocol*
TIC	Tecnologias de Informação e Comunicação
TLS	*Transport Layer Security*
TMN	*Telecomunication Management Network*
TSDB	*Time Series Database*
UDP	*User Datagram Protocol*
UML	*Unified Modelling Language*
UO	Unidades Organizacionais
U. Porto	Universidade do Porto
UPdigital	Universidade do Porto Digital
UPS	*Uninterruptible Power Supply*
UPTEC	Parque de Ciência e Tecnologia da Universidade do Porto
USM	*User-Based Security Model*
VCAM	*View-Based Access Control Model*
VCS	*Version Control System*
VIP	*Virtual IP Address*
VLAN	*Virtual Local Area Network*
VOIP	*Voice over Internet Protocol*
VPLS	*Virtual Private LAN Service*
VPN	*Virtual Private Networks*
VRF	*Virtual Routing and Forwarding*
VRRP	*Virtual Router Redundancy Protocol*
WBEM	*Web Based Enterprise Management*
Wifi	*Wireless Fidelity*
WinRM	*Windows Remote Management*
WLC	*Wireless LAN Controller*

WMI	*Windows Management Instrumentation*
WSF	*Work Station Function*
wsrep	*Write-set replication*
XML	*eXtensible Markup Language*
Xtra	*Extra*
ZPA	Zabbix *proxy Access*
ZPCD	Zabbix *proxy Core and Distribution*
ZS	Zabbix *Servers*

Capítulo 1

1 Introdução

1.1 Enquadramento

As redes de comunicação de dados estão em constante evolução, proporcionando vários desafios, nomeadamente, as operações de manutenção e gestão das infraestruturas de comunicação e serviços. Para além disso, o crescimento de uma rede, origina a dependência de um maior número de serviços por si prestados. Tudo isto são desafios complexos de resolver, sem recorrer a uma ferramenta de monitorização de redes. A principal função destas aplicações consiste na monitorização de equipamentos ativos, bem como dos serviços por estes disponibilizados, com recursos a verificações periódicas.

De um modo habitual, todas as instituições dependentes de recursos da sua rede, devem utilizar mecanismos de monitorização e alarmística, que de forma centralizada, enviem alertas de acordo com o estado da rede.

A rede da Universidade do Porto (U. Porto) é um exemplo de uma instituição, que recorre a este tipo de ferramentas de monitorização. A grande dependência da monitorização, tanto na vertente alarmística, quanto na vertente estatística, prende-se pela extensão dos seus equipamentos da camada de núcleo e os seus múltiplos serviços intrínsecos. Como base, todo o *core* da rede de comunicação da Universidade do Porto, assenta em quatro *routers* distribuídos por diferentes salas técnicas, instaladas em edifícios localizados na Reitoria, na Faculdade de Ciências, na Faculdade de Engenharia e na Faculdade de Direito. Estes equipamentos encontram-se interligados por circuitos de fibra ótica implementados numa tipólogia em anel permitindo a redundância de caminhos e a interligação de cinquenta e sete nós de rede (instalados nas diferentes Faculdades, Residências Universitárias e Institutos de Investigação, que pertencem organicamente à U. Porto).

De forma a disponibilizar o acesso à *Internet* para a Universidade do Porto, existem também dois circuitos de fibras que interligam os *routers* localizados na FEUP (Faculdade de Engenharia da Universidade do Porto) e na FCUP (Faculdade de Ciências da Universidade do Porto), à RCTS (Rede, Ciência, Tecnologia e Sociedade). Dentro deste domínio de rede,

existem também diversos equipamentos ativos, nomeadamente, *switches*, *firewalls*, servidores, controladoras *wireless*, entre outros.

Devido à dimensão de *backbone* da Universidade do Porto, existe uma solução de rede baseada em VPLS (*Virtual Private LAN Service*) sobre IP/MPLS (*Internet Protocol*/*Multi Protocol Label Switching*) para a disponibilização de serviços de conectividade em *Ethernet*.

No processo de gestão e monitorização de redes, é utilizado na componente alarmística uma solução baseada em Nagios e, no processo de monitorização estatística, é utilizada uma aplicação denominada Grafana.

Contudo, com a evolução da rede, e de forma a garantir a escalabilidade da mesma, começa a surgir um conjunto de desafios que levam à necessidade de explorar novas ferramentas de monitorização alarmística e, assim, promover aspetos fundamentais, que não obrigue a despender de tanta intervenção humana, como por exemplo, o processo adicionar ou remover equipamentos, nas plataformas de gestão de rede. Em paralelo, cada vez mais, as ferramentas de monitorização permitem de forma centralizada ter uma visão topológica da rede, tanto ao nível de tráfego, quanto ao nível da rápida identificação de problemas que possam surgir.

Assim, o desafio passa por estudar a melhor solução para estes problemas e, ao mesmo tempo, criar uma arquitetura transversal à rede da Universidade do Porto, ou seja, que funcione de forma transparente e centralizada, integrando uma monitorização ao nível do serviço e adequando-se a uma plataforma já existente na componente de monitorização estatística (Grafana).

1.2 Objetivos e motivações

Com este projeto pretende-se propôr uma possível solução de monitorização e gestão de redes de comunicação, sistemas e serviços, destinada à Universidade do Porto. Os principais objetivos que se pretendem alcançar com a elaboração deste projeto são:

- Identificar e mitigar as limitações dos sistemas de monitorização FCAPS que se encontram em produção na Universidade do Porto;
- Analisar as aplicações *open-source* mais relevantes no mercado, para a monitorização alarmística;
- Implementar as plataformas de monitorização num ambiente em operação, para prova de conceito, identificando vantagens e desvantagens e promovendo assim as mais-valias para a Universidade do Porto;

É importante salientar que existem requisitos pré-estabelecidos pela Universidade do Porto, sendo de certa forma, objetivos a atingir, mas que irão influenciar mais na escolha da ferramenta de monitorização alarmística a adotar. Enumeram-se de seguida esses requisitos:

- A ferramenta adotada deve suportar a monitorização de serviços de *Layer-2* (*Ethernet*), prestados pela infraestrura IP/MPLS, os quais são utilizados pelas Unidades Orgânicas da Universidade do Porto para diversos ambientes de conectividade;
- Criar uma solução que permita otimizar as tarefas, no processo de adicionar um novo dispositivo de rede, tanto ao nível de *hardware* como de *software*. Este conceito é definido por "autodescoberta";
- Garantir o armazenamento resiliente dos dados recolhidos pelo sistema de monitorização adotado;
- Integração da infraestrutura na rede de *backbone* da Universidade do Porto, garantindo a sua redundância aplicacional e de localização geográfica;
- O *software* de monitorização alarmística terá de ser configurado e aprovisionado, de forma a dar suporte a diferentes equipamentos e tecnologias (*firewalls, routing*, MPLS);
- Estudar uma solução centralizada que seja escalável com a rede de *Core* da Universidade do Porto, adequando-se à gestão técnica e operacional.

A motivação para realizar este trabalho adveio sobretudo de uma intenção pessoal, paralelamente com necessidades profissionais. Através do desempenho de funções na Unidade de Serviços de Rede Centrais e *Datacenters*, e com a prestação de serviços à UPDigital, a investigação da componente de monitorização alarmística, irá dotar o autor, de um aumento de conhecimento na área das redes de comunicação de dados, bem como na vertente de administração de sistemas. Estes fatores são preponderantes, tanto para o desenvolvimento do conhecimento académico, como também para satisfazer as necessidades e desafios tecnológicos da própria Universidade do Porto.

1.3 Metodologia

Para dar resposta adequada aos objetivos elencados no ponto anterior, este trabalho assentou numa investigação empírica (revisão bibliográfica), em experimentação (ambiente de testes) e num estudo comparativo, tendo por base os seguintes pontos:

a. Analisar o estado atual da rede da U. Porto e as suas necessidades na componente de monitorização;
b. Investigar aplicações de monitorização (*open source*) disponíveis no mercado;
c. Analisar os paradigmas de gestão e monitorização de redes na pesquisa do item b);
d. Analisar os paradigmas de técnicas de armazenamento na pesquisa do item b);
e. Demonstrar e/ou testar a utilização das ferramentas em ambiente de testes;
f. Analisar comparativamente os resultados apresentados;
g. Desenvolver a solução, e todos os seus componentes de encaminhamento, armazenamento e consulta, adequados às necessidades da U. Porto com base nos resultados do item f);
h. Apresentar a análise geral dos resultados obtidos e tecer comentários sobre este projeto.

Utiliza-se, além da pesquisa bibliográfica, conceitos quantitativos e qualitativos. Os conceitos quantitativos são usados para avaliar os dados pesquisados para posterior análise. É também um trabalho qualitativo, pois analisa a qualidade funcional e a usabilidade de várias aplicações *open source*.

Quanto ao ponto de vista da natureza da pesquisa, esta pode ser classificada como pesquisa aplicada, pois visa gerar conhecimento para apresentar uma solução prática de um problema específico [1].

1.4 Estrutura do documento

O documento está organizado em seis capítulos e vinte e oito anexos, principiando pela introdução, que se apresenta neste capítulo, com a descrição do enquadramento, dos objetivos e motivações do presente relatório, a metodologia utilizada e esta estrutura.

A revisão bibliográfica, ou estado de arte, é contextualizada no capítulo 2. Esta contextualização discorrerá sobre a área funcional de gestão de redes de comunicação. Destaca-se também, neste capítulo, as Arquiteturas OSI (*Open System Interconnection*) e TCP/IP (*Transmission Control Protocol*), com abordagens simples e pragmáticas para o seu normal funcionamento, tendo em vista compreender a metodologia utilizada nos modelos de gestão, também aqui descritos, e as plataformas de gestão de rede, que fornecem funcionalidades a diversos níveis, integrando diferentes ambientes protocolares e permitindo uma melhor gestão operacional das redes de grande dimensão.

O capítulo 3 descreve o estado atual da rede da Universidade do Porto, analisando a sua infraestrutura e o plano de gestão operacional, que visa conceptualizar métodos funcionais, sejam eles recursos humanos, materiais ou tecnológicos, para gerir e administrar uma organização de forma eficaz, sem comprometer a sua produtividade.

O estudo de ferramentas de gestão de redes é apresentado no capítulo 4. Mais especificamente, descreveu-se o ambiente de testes, quais os critérios de seleção das ferramentas/plataformas estudadas, de acordo com a análise de experimentação, objetivando as mais e menos valias de cada uma e, por fim, a análise comparativa destas e os resultados da mesma.

Em seguida, é apresentado o capítulo 5. Este foca-se na solução para o sistema de monitorização alarmística/estatística, nomeadamente, as componentes que a integram, as especificações das aplicações utilizadas e seus requisitos, e no seu funcionamento genérico. Abordou-se ainda, a arquitetura de armazenamento de dados e dos servidores de monitorização, propôs-se melhorias da topologia da solução e, finalmente, descreveu-se a integração da solução na infraestrutura de rede de dados.

O capítulo 6 apresenta a conclusão deste relatório. São apresentados os resultados obtidos, é efetuada a discussão dos mesmos, bem como propostas para desenvolvimentos futuros.

Capítulo 2

2 Gestão de redes

2.1 Introdução

A abrangência, a diversidade de sistemas e a complexidade das redes atuais, exige o planeamento de uma rede de gestão informática, considerando um conjunto de mecanismos e ferramentas para monitorização e controlo dos recursos de comunicação.

A dimensão dos serviços suportados pelas redes desencadeia necessidades de resposta em termos de controlo, coordenação e monitorização, de modo a possibilitar a oferta de serviços com a qualidade exigida. Preferencialmente, essas necessidades requerem uma forma de gestão integrada, que possibilite o acesso à informação de rede a diferentes níveis.

O conceito de gestão de redes ganha portanto bastante relevo no processo de desenvolvimento e manutenção de uma rede. O gestor de rede necessita de informação sobre o seu funcionamento de rede, que lhe permita, por exemplo, negociar níveis de qualidade de serviço, tomar opções de funcionamento ou, ainda, ser informado atempadamente de possíveis falhas de comunicação entre os equipamentos e os serviços de rede.

Por todos estes aspetos, torna-se imprescindível a necessidade de integrar, de algum modo, nos sistemas de comunicação, ferramentas para recolher, transferir, arquivar, analisar e apresentar a informação de gestão de rede. Estas ferramentas têm por base uma arquitetura suportada no conceito gestor-agente, que recorre a múltiplos protocolos, baseados em modelos e paradigmas de gestão de redes, nomeadamente, os modelos organizacionais, de comunicação, de informação e funcionais.

Tendo em conta a evolução tecnológica, as estratégias para a gestão de redes têm progredido desde a utilização de aplicações integradas nos sistemas de comunicação. Este processo, maioritariamente, recorre a protocolos de gestão, como por exemplo, o protocolo SNMP (*Simple Network Management Protocol*). No entanto, surgem novas plataformas de gestão baseadas em aplicações *web*, proprietárias ou de código aberto, que permitem novas funcionalidades de gestão, utilizando também, novos protocolos e linguagens de programação.

Com o cenário acima apresentado, pretende-se, nas secções seguintes, identificar o modelo de referência OSI (*Open System Interconnection*) e o seu modelo sucessor TCP/IP. De seguida, apresentam-se os modelos e paradigmas de gestão, subjacentes às diversas soluções de gestão existentes. Por fim, enumeram-se os tipos de gestão, em redes de telecomunicações, baseada em aplicações *web* ou sustentada em políticas.

2.2 Arquiteturas OSI e TCP/IP

As atividades de gestão de redes podem ser compreendidas aos mais diversos níveis, abrangendo aspetos que vão desde a monitorização de simples elementos de rede, até à gestão de redes mais complexas. A gestão dos serviços e aplicações distribuídas em rede necessitam de modelos arquitetónicos (OSI e TCP/IP) com abordagens simples e pragmáticas para o seu normal funcionamento. No entanto, é necessário compreender a metodologia utilizada nos modelos de gestão OSI e TCP/IP, de modo a estabelecer conceitos transversais, e assim entender as necessidades de uma rede de gestão baseada em camadas protocolares. Todos os protocolos de gestão de redes assentam então, numa arquitetura indispensável ao seu normal funcionamento [2].

Numa primeira abordagem surgiu o modelo de referência OSI, como resultado de um projeto desenvolvido pela ISO durante os anos 70 e 80. Este modelo surge com o objetivo de definir um conjunto de conceitos para o desenvolvimento de protocolos e serviços de comunicação concretos. Este modelo agrupa as funcionalidades de comunicação em sete camadas, de acordo com critérios de afinidade, abrangendo aspetos que vão desde o equipamento de interface com os meios físicos, até aos protocolos de aplicação [3].

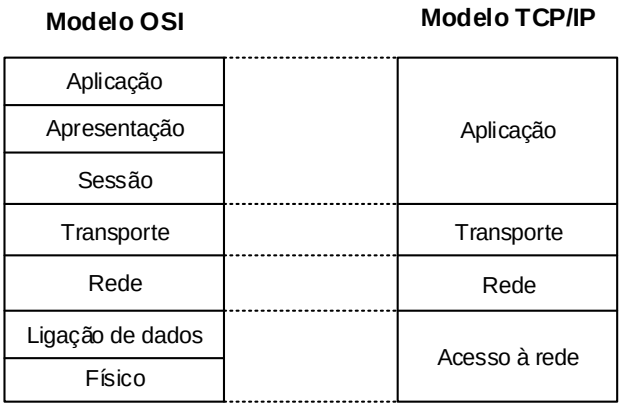


Figura 1 - Níveis de arquiteturas protocolares OSI e TCP/IP

Posteriormente, como resultado de uma atitude prática em relação ao problema de comunicação fiável entre os componentes de rede, a arquitetura TCP/IP originou soluções menos complexas face às necessidades existentes. Esta arquitetura protocolar é composta por quatro camadas, em vez das sete camadas preconizadas pelo modelo OSI. A Figura 1 representa a arquitetura protocolar TCP/IP, em comparação à arquitetura OSI, onde é visível a correspondência entre as camadas constituintes, bem como a aglutinação de camadas. Em seguida, descrevem-se as diversas camadas constituintes da arquitetura protocolar TCP/IP.

O nível de acesso à rede engloba a camada física (*layer* 1) e a camada de ligação de dados (*layer* 2) do modelo OSI. Esta camada constitui uma interface entre o meio físico de comunicação (exemplo, corrente elétrica), estabelecendo uma forma de representação lógica da informação (*bits* de valor lógico, 0 ou 1). Por sua vez, a camada de acesso à rede também tem como objetivo a garantia de comunicação, podendo fornecer mecanismos de controlo de fluxo de informação e controlo de erros. Esta camada trabalha com conjuntos de *bits* organizados em tramas.

Algumas das principais funções desta camada são o encapsulamento de pacotes IP nas tramas a transmitir posteriormente para a rede e a tradução de endereços IP em endereços *Ethernet* (*Media Access Control*, MAC), com recurso ao protocolo ARP (*Address Resolution Protocol*) [4].

A camada de rede (*layer* 3) é, também, designada por nível de *internet*, o qual, recorre a uma arquitetura protocolar (IP). Este nível é responsável pelo direcionamento de pacotes na rede, executando o processo de encaminhamento (*routing*) com base nos endereços IP de destino. Neste nível, são efetuadas ações de fragmentação e reassemblagem de pacotes, ajustando-se o tamanho máximo das unidades de informação suportados pela camada subjacente. Esta camada utiliza protocolos como o ICMP (*Internet Control Message Protocol*) [5] e o IGMP (*Internet Group Management Protocol*) [6].

A comunicação realizada extremo a extremo é assegurada pela camada de transporte (*layer* 4). O nível de transporte recorre a dois protocolos, nomeadamente o UDP (*User Datagram Protocol*) [7] e o TCP (*Transmission Control Protocol*) [8], utilizando-se nesta camada o conceito de portos de origem e destino, estabelecendo assim uma ligação lógica. O protocolo UDP funciona em modo de ausência de ligação, não garantindo a transferência fiável de informação. Por outro lado, o protocolo TCP é funcionalmente mais rico, operando em modo de ligação e garantindo a transferência livre de erros de qualquer fluxo de informação entre o

emissor e o recetor.

Por fim, o nível de aplicação (*layer* 5) corresponde às camadas de sessão, apresentação e aplicação do modelo OSI. A camada aplicacional do modelo TCP/IP oferece serviços que interessam diretamente aos utilizadores ou a processos de aplicação. Existe uma grande variedade de protocolos de aplicação, correspondendo à grande diversidade de necessidades dos utilizadores, destacando-se os protocolos de utilização mais comuns, nomeadamente:

- Telnet – Protocolo de terminal virtual;
- FTP – Protocolo para acesso e transferência de ficheiros;
- SMTP – Protocolo de envio correio eletrónico;
- HTTP – Protocolo de hipertexto/hipermédia;
- DNS – Sistema de mapeamento entre nomes e endereços IP;
- SNMP – Protocolo para suporte de aplicações e gestão de redes.

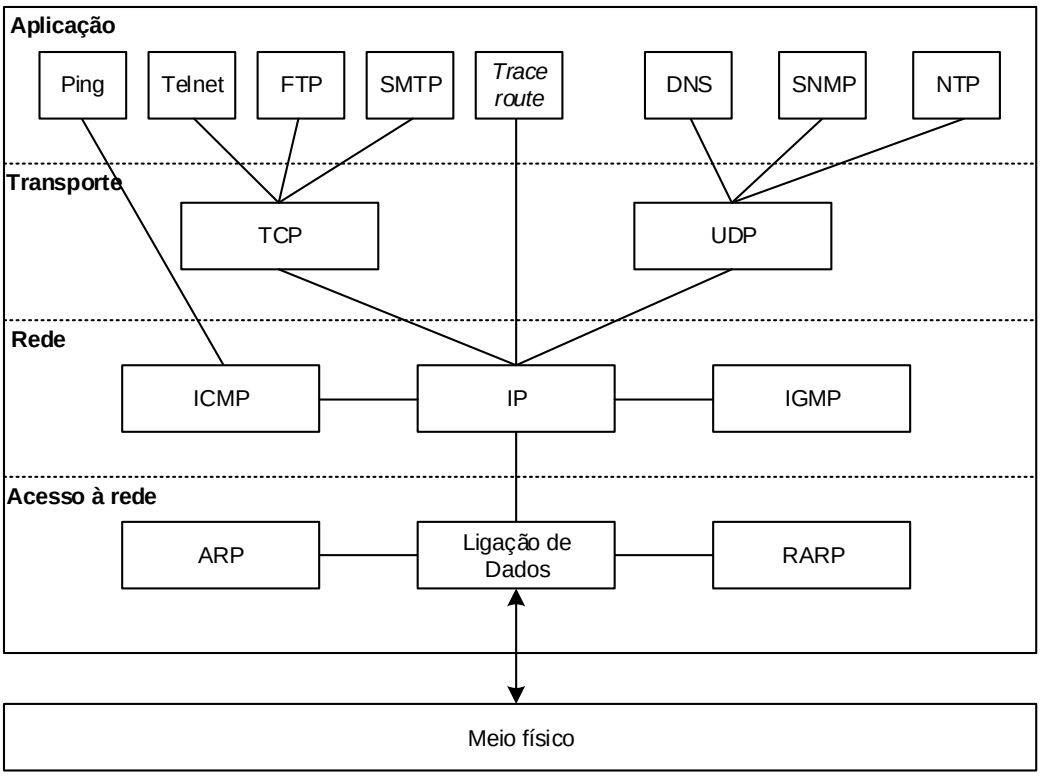


Figura 2 - Localização de diversos protocolos da arquitetura TCP/IP

A Figura 2 ilustra o posicionamento dos protocolos apresentados anteriormente e de outros protocolos vulgarmente utilizados na arquitetura TCP/IP.

2.3 Modelos e paradigmas para gestão de redes

Existem vários modelos e paradigmas de gestão de redes (Anexo III), sendo que nas redes de comunicação de dados é utilizado como referência, o modelo gestor-agente, representado na Figura 3.

Um sistema de gestão de rede abrange uma entidade de gestão, denominada por gestor, responsável por executar as ações de gestão, propriamente ditas. Por sua vez, os sistemas geridos ou agentes são elementos de rede que recorrem a um protocolo de gestão, de modo a executar a comunicação com o gestor e assim acederem à informação de gestão neles armazenada. Um ou mais sistemas geridos possuem agentes que enviam e recebem notificações do gestor, tendo estes também a função de armazenar numa base de dados, toda a informação relativa aos objetos geridos [3].

Quando um elemento de rede não implementa um protocolo de gestão, é utilizado o conceito de *proxy*, deste modo, os dispositivos interligados a um sistema *proxy*, não necessitam de agente. Esta solução é tipicamente utilizada em sistemas que não suportavam protocolos de gestão [3].

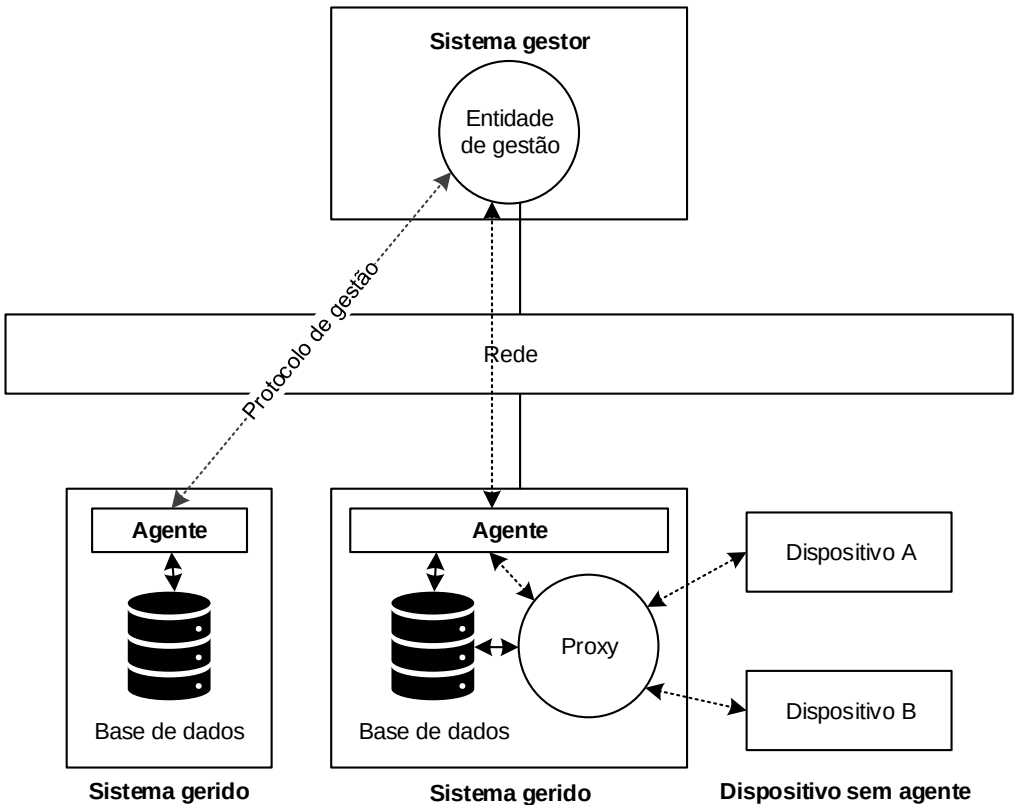


Figura 3 - Modelo genérico gestor-agente utilizado nos sistemas de gestão

A gestão de redes pode ser vista de diversos pontos de vista, no entanto existem submodelos que permitem segmentar as arquiteturas de gestão de redes, e assim definir por módulos, todas as funções inerentes ao seu bom funcionamento. Os quatro submodelos fundamentais que decompõem uma arquitetura de gestão são os seguintes [9]:

- Modelo organizacional - institui os domínios de gestão e delimita responsabilidades dentro dos domínios organizacionais. No modelo organizacional, os objetos geridos são distribuídos por diversos domínios, sendo estabelecidas as responsabilidades e os papéis do objeto gerido (agente) e da entidade de gestão (gestor);
- Modelo de comunicação - estabelece os mecanismos para a troca de informação de gestão, mais concretamente, as atividades de configuração de objetos geridos, o seu estado e a comunicação de notificações. Neste modelo é também definido os protocolos e serviços disponíveis para suportar as atividades anteriormente referidas, com base numa sintaxe e semântica de comunicação;
- Modelo de informação - relaciona aspetos descritivos dos objetos geridos (agentes) com os recursos a gerir (componentes, protocolos ligações virtuais, etc.). Todos os modelos de informação incluem uma descrição sintática e semântica dos objetos geridos, como por exemplo, a definição do comportamento dos objetos, operações executáveis sobre os mesmos, identificação dos objetos existentes, bem como, as suas propriedades e relações, e o mapeamento entre a informação de gestão guardada nas bases de dados (*Management Information Base*, MIB);
- Modelo funcional - especifica as funções de gestão e a forma como estas se inter-relacionam, de modo a fornecer a finalidade pretendida. Em cada uma das áreas funcionais (identificadas na secção 2.3.4), este modelo define as funcionalidades esperadas, os serviços e funções necessárias para cada área funcional, e também, os objetos relevantes a serem geridos.

Na Figura 4, é possível identificar os principais submodelos de uma arquitetura de gestão de redes, e os respetivos elementos que cada um deles abrange.

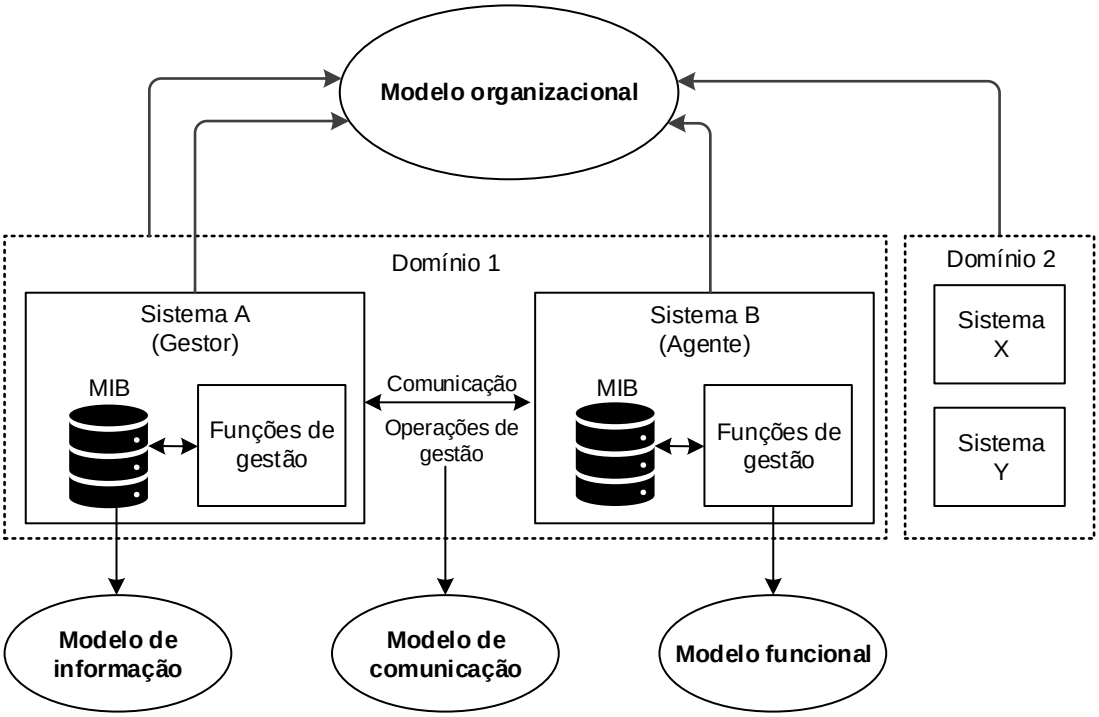


Figura 4 - Submodelos de gestão

2.3.1 Modelo organizacional

A arquitetura de gestão OSI define dois papéis possíveis para as entidades de gestão: o papel de gestor e o de agente. O gestor e o agente relacionam-se através da utilização de protocolos de gestão, possibilitando assim, a execução de operações sobre os objetos geridos, obter os resultados das operações solicitadas, obter mensagens de erro e gerar ou receber notificações. Mediante certas operações, poderá ser atribuído de forma dinâmica o papel de gestor e de agente num dado sistema [3].

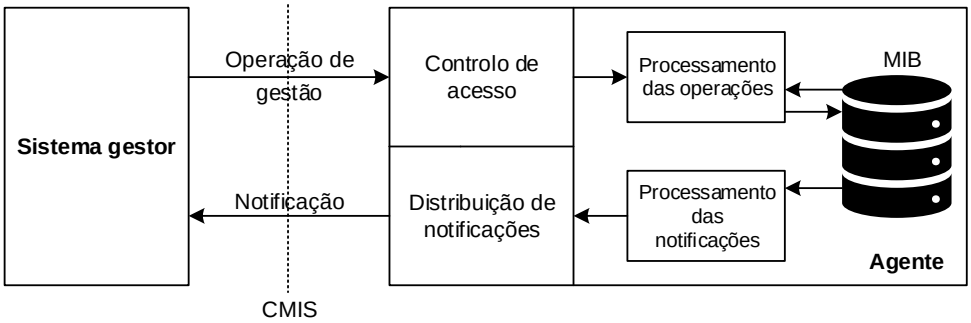


Figura 5 - Arquitetura gestor-agente no modelo de gestão OSI

A Figura 5 ilustra as interações entre um sistema gestor e um sistema agente. Este processo de relacionamento entre os dois sistemas é realizado com recurso à utilização de um serviço de informação de gestão, comum aos dois sistemas (*Common Management Information Service*, CMIS) [10].

Com a evolução da arquitetura de gestão OSI para a arquitetura de gestão TCP/IP, o paradigma destas arquitetura gestor-agente ficou semelhante. A diferença mais significativa foi realizada ao nível do processo de comunicação (consultar secção 2.3.2), entre os sistemas de gestor e agente.

2.3.2 Modelo de comunicação

A arquitetura de gestão OSI define três áreas referentes ao modelo de comunicação [3]:

- Gestão de sistema (*system management*) – entidade responsável por ações de gestão, referentes a um elemento de rede. Os processos de aplicação de gestão de sistema *(Systems Management Application Processes*, SMAP) comunicam-se com base em entidades de aplicação de gestão de sistema (*Systems Management Application Entities*, SMAE), recorrendo a serviços comuns de informação de gestão (*Common Management Information Services*, CMIS) [10]. Com base num protocolo comum para informação de gestão (*Common Management Information Protocols*, CMIP), são definidos procedimentos para a transmissão de informação de gestão e sintaxe, para os serviços comuns de informação de gestão CMIS;

- Gestão de camada (*layer management*) – reúne funções, serviços e protocolos específicos de uma dada camada, sendo esta suportada por entidades de gestão de camada (*Layer Management Entities*, LME) e protocolos de gestão de camada [11];

- Operação de camada (*layer operation*) – compreende as funções de controlo e operação dentro dos protocolos de camada. Nestas funções podemos destacar como exemplo, o controlo de erros, controlo de fluxo e controlo de sequência.

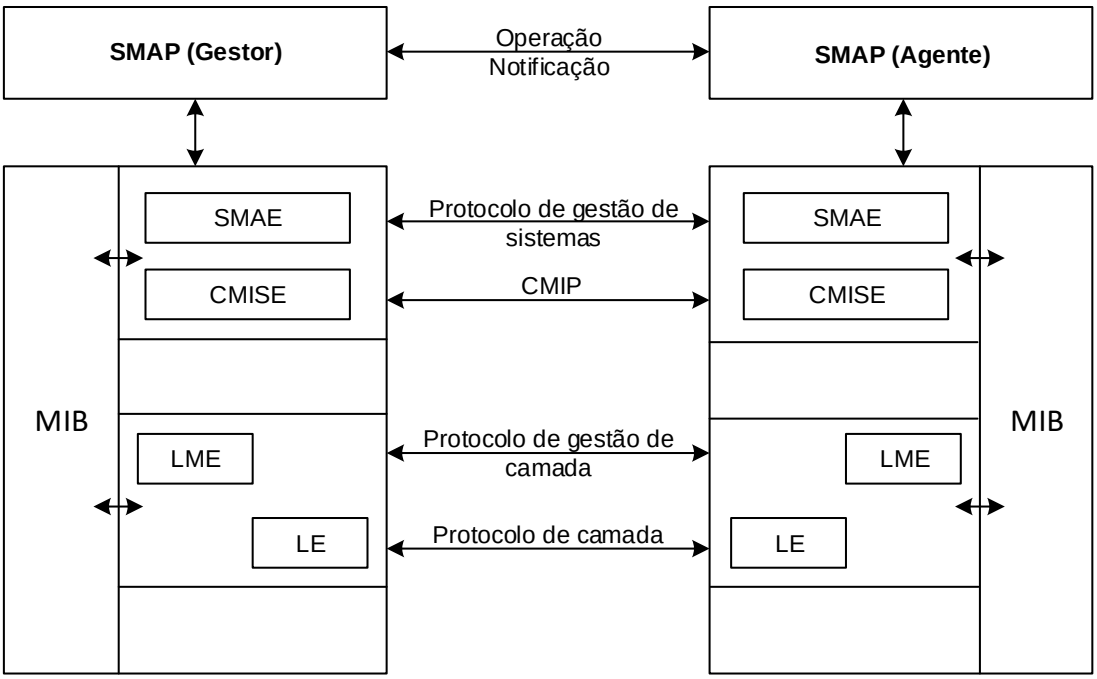


Figura 6 - Modelo de comunicação OSI

O protocolo CMIP encontra-se normalizado na *International Organization for Standardization* e era utilizado na arquitetura de gestão OSI. Com a evolução da arquitetura de gestão OSI para a arquitetura de gestão da *Internet*, foi desenvolvido o protocolo CMOT, *Common Management Information Protocol*, (CMIP *over* TCP/IP), de forma a tentar introduzir o protocolo CMIP, na arquitetura de gestão TCP/IP. O protocolo CMOT apresentava muita complexidade de configuração [12], o que de certa forma, beneficiou a adoção de um protocolo simples de gestão de redes, nomeadamente o *Simple Network Management Protocol*, (SNMP).

O protocolo SNMP tornou-se então, o protocolo mais utilizado na arquitetura de gestão de redes TCP/IP, mesmo apresentando algumas lacunas no seu modelo de funcionamento inicial.

Este protocolo foi desenvolvido com base no protocolo SGMP *(Simple Gateway Monitoring Protocol*), que surgiu em 1987. As limitações do protocolo SGMP (apenas monitorizava *Gateways*) [13], originaram o desenvolvimento de um novo, o SNMP, o qual foi definido na série de *standards* da IETF (*Internet Engineering Task Force*), e não só este protocolo, mas também a linguagem que estrutura as base de dados (*Management Information Base*, MIB). A linguagem que define as MIB's é denominada por *Structure of Management Information* (SMI), contemplando duas versões (SMIv1, SMIv2) [14], [15].

No modo de funcionamento, o protocolo SNMP, faz uso da camada aplicacional *(application layer*) da arquitetura TCP/IP, sobre o protocolo de transporte UDP (*User Datagram Protocol*), o qual não garante a entrega dos pacotes IP, mas possibilita que as operações de gestão sejam tão leves e eficientes quanto o possível [3].

As operações do protocolo SNMP são realizadas em bases de dados MIB's. A informação de gestão das MIB's é definida numa hierarquia de objetos, onde cada um possui um identificador exclusivo (*Object Identifier*, OID).

O protocolo SNMP apresenta atualmente três versões que contemplam resumidamente as seguintes características:

- SNMPv1 – define a primeira versão do SNMP, e dá suporte a quatro operações básicas, nomeadamente as operações *Get-Request*, *Get-Next-Request*, *Set-Request* e *Trap*. Para além de suportar apenas quatro operações, existem problemas de ineficiência e escalabilidade, no que concerne à atualização de grandes volumes de dados, nomeadamente o excesso de tráfego de gestão. Outro aspeto negativo do SNMPv1 é a segurança, praticamente inexistente [16];

- SNMPv2 – comporta melhorias em termos do modelo de informação (MIB), de comunicação e de segurança. Esta versão foi padronizada em 1996 pela IETF e encontra-se definida nas RFCs 1441 a 1452. O protocolo SNMPv2 ganhou relevo, pelas suas novas potencialidades com o incremento de duas novas operações. A operação *Get-Bulk-Request* possibilita a leitura de uma tabela inteira, ao contrário do que acontecia no SNMPv1. A segunda operação implementada foi o *Inform-Request*, que introduziu o conceito de papel duplo gestor-agente e, deste modo, em certas interações, uma entidade que desempenha o papel de gestor, pode também simultaneamente ser agente.

  Outra vantagem foi a interação no processo de troca de informação de gestão entre dois sistemas gestores (gestor-gestor) [17];

- SNMPv3 – adicionou segurança às versões anteriores do SNMP. Concretamente, foram acrescentados métodos de autenticação, encriptação, privacidade, autorização e controlo de acesso, das ferramentas administrativas, por exemplo, gestão de utilizadores e chaves, configuração remota via SNMP, interoperabilidade com *proxy servers* [18].

Arquitetura genérica do protocolo SNMP

Existem alguns componentes fundamentais que decompõem a arquitetura SNMP, conforme ilustra a Figura 7. O gestor (*Network Management System*, NMS) contém uma aplicação que efetua a recolha e o processamento de dados relativos à informação de gestão, de um dado dispositivo.

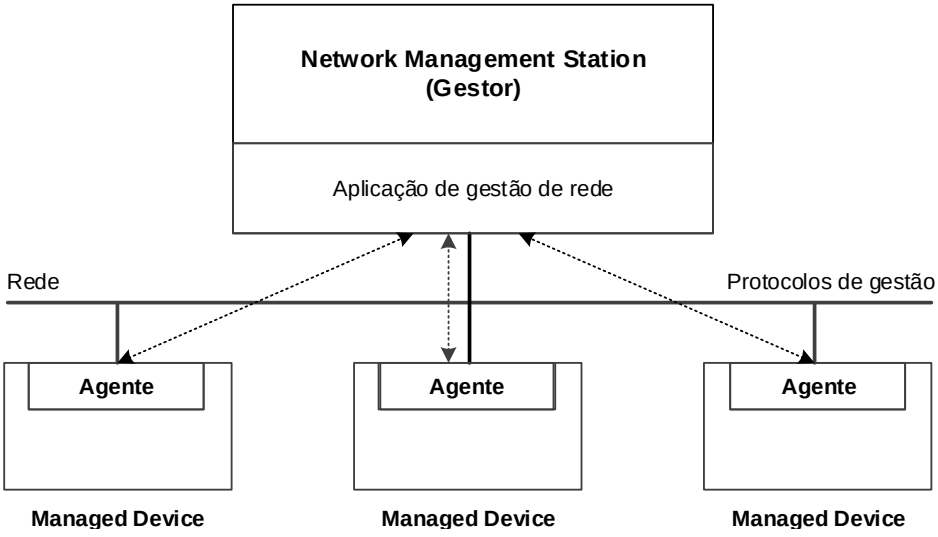


Figura 7 - Modelo genérico de gestão TCP/IP

O módulo de *software* de gestão de rede, presente num dipositivo, que envia e recebe informação de gestão proveniente do gestor (NMS), é denominado por agente. Este tem como principal função, traduzir os dados armazenados na base de dados (MIB) para a linguagem compreendida pelo SNMP (SMI). Todos os detalhes protocolares do SNMP, ao nível de operações e formatos das mensagens nas diversas versões, são retratados em pormenor no estudo inerente ao Anexo I.

Todos os equipamentos que possuem um agente, são também nomenclados de *Managed Device* (MD). Por sua vez, os agentes possuem uma MIB, onde existe um conjunto de objetos estruturados que seguem uma hierarquia. Estes objetos, como referido anteriormente, são compostos por nome, sintaxe e OID.

2.3.3 Modelo de informação

. O documento da ISO 10165 (*Structure of Management Information*, SMI) [19] formaliza a estrutura do modelo de informação de gestão, que utiliza como sintaxe abstrata o ASN.1 (*Abstract Syntax Notation One*).

Numa primeira abordagem o modelo de informação foi baseado na arquitetura de gestão OSI. O conceito de modelo de informação surge com a base de dados MIB definida pela ISO. Este modelo baseia-se numa aproximação orientada a objetos, para o conceito de objetos geridos. Os objetos geridos são definidos em termos dos seus atributos e das operações que podem ser executadas sobre eles.

Ao contrário da aproximação da MIB definida pelo OSI, a arquitetura de gestão TCP/IP não usa o paradigma *object-oriented*, embora use alguns conceitos próximos, o que conduz a uma solução mais simples e fácil de entender, não sendo, no entanto, tão poderosa.

MIB - *Management Information Base*

O modelo de informação foi definido numa estrutura genérica que permite a identificação da informação de gestão, recorrendo a uma árvore de registo com elementos numa linguagem utilizada para descrever a informação de gestão [14].

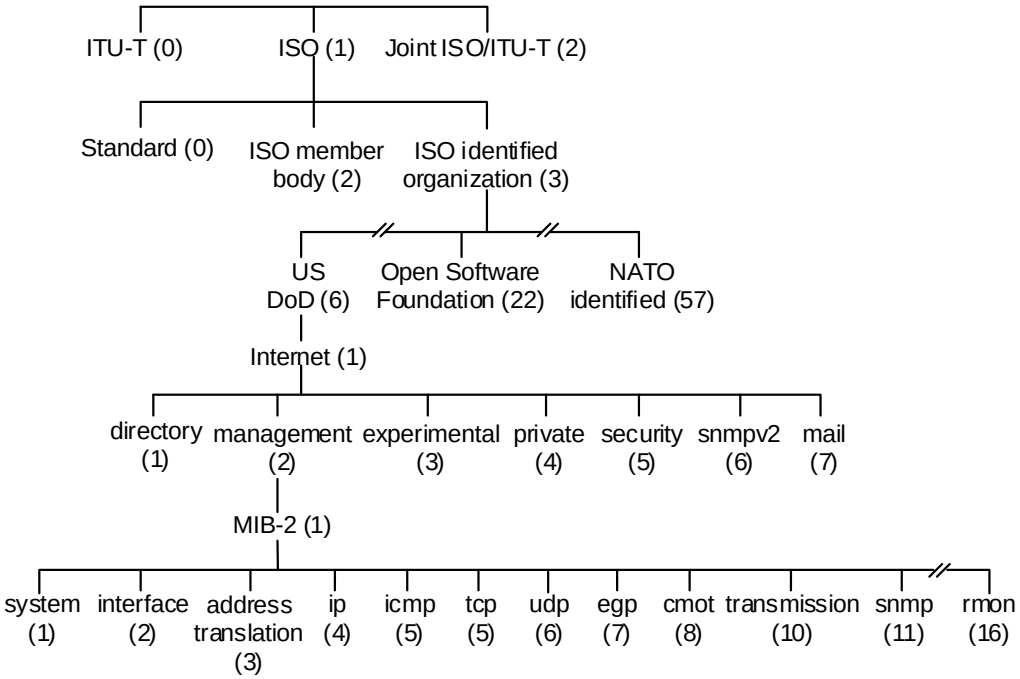


Figura 8 - Árvore de identificadores de objetos

Para possibilitar a identificação unívoca de toda a informação de gestão, é utilizada uma árvore de nomeação, que permite a definição de identificadores de objetos (OID). Um objeto existente numa árvore de nomeação contém um identificador único, o qual obtém-se concatenando os números dos nós da árvore, desde a raiz até ao nó em causa [12].

A Figura 8 representa a parte da árvore de identificadores que contém objetos relativos à gestão de rede, podendo estes serem identificados pelo prefixo 1.3.6.1.2.

Toda a informação de gestão é então organizada em bases de informação de gestão. Conceptualmente uma MIB é um repositório de dados que reside nos agentes, que é acedido pelo gestor, usando um protocolo de gestão (SNMP).

A versão inicial MIB (MIB-I) continha um limite predeterminado de cem objetos [14]. Com a necessidade de introdução de novas variáveis surgiu a MIB-II. Os objetos definidos na MIB-II são cerca de cento e setenta e organizam-se em dez grupos distintos [20], mormente:

- Grupo sistema (*system*) – possui informação de configuração do nó da rede onde se encontra a MIB, nomeadamente, informação acerca do responsável pelo sistema e informação sobre os serviços implementados no dispositivo;
- Grupo de interfaces (*interfaces*) – disponibiliza informação relativa às interfaces do sistema, particularmente, tipo e descrição das mesmas, informação de configuração e informação acerca do estado;
- Grupo de tradução de endereços (*address tranlation*) – contém informação disponibilizada pelo processo de resolução de endereços; Os seis grupos alusivos a protocolos, nomeadamente IP, ICMP, TCP, UDP, EGP (*Exterior Gateway Protocol*), que possuem contadores de unidades de dados respetivos a entradas e saídas, bem como, quantificadores de erros, e tabelas com informações características de cada protocolo;
- Grupo de transmissão (*transmission*) – estabelece um conjunto de objetos referentes a interfaces de rede específicas.

***Remote Network Monitoring* MIB**

O repositório de dados *Remote Network Monitoring* MIB (RMON MIB), permite alocar a informação sobre o estado da rede, ao contrário da MIB tradicional que recolhe informação dos sistemas [21]. A RMON MIB pertence à árvore de identificadores de objetos conforme

demonstra a Figura 8, e é definida pelo prefixo 1.3.6.1.2.1.16.

Embora os objetos existentes na RMON MIB contenham uma complexidade superior aos objetos da MIB, o paradigma de gestor-agente deixa de ser utilizado, conforme ilustra a Figura 9. Deste modo os sistemas geridos que utilizam a RMON MIB são dispositivos com uma funcionalidade de gestão própria, como por exemplo analisadores de protocolos, que processam a informação de gestão e a armazenam [3].

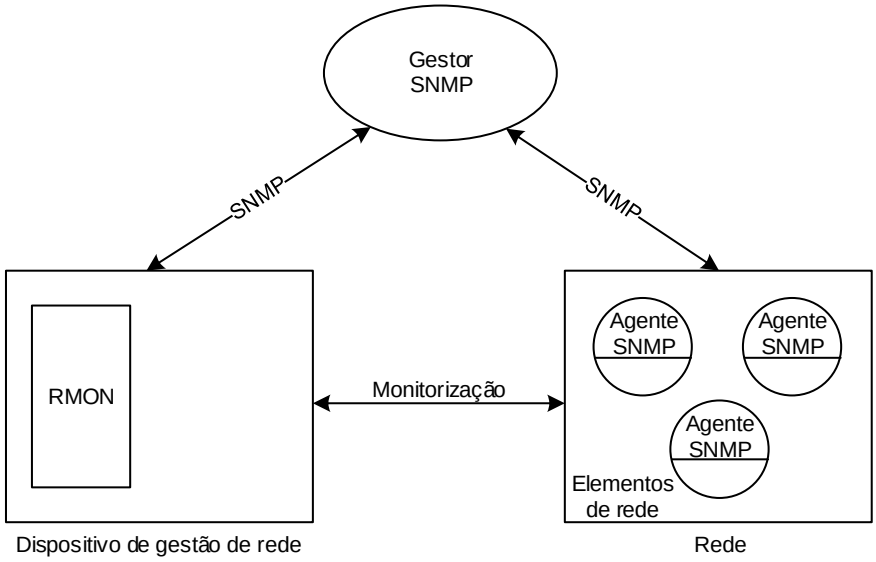


Figura 9 - Processo de gestão RMON

O RMON MIB advém, fundamentalmente, de uma tecnologia de monitorização e gestão de redes locais (*Local Area Network*, LAN), não contendo objetos de nível superior à camada de ligação de dados do modelo de referência OSI. Esta limitação foi resolvida com o repositório de dados RMON2 MIB, que suporta objetos das camadas protocolares superiores, permitindo assim que a gestão de redes se estenda para além das redes locais [22]. Os grupos de objetos do RMON MIB e RMON2 MIB disponibilizam diversas informações, tal como é demonstrado na Tabela 1 e na Tabela 2 [21] [23].

A utilização destes objetos permite o pré-processamento de informação no dispositivo local e a sua posterior entrega, de forma automática, através da utilização de um mecanismo de geração de eventos.

Tabela 1 - Grupo de objetos RMON MIB

Grupo RMON	Função
Statistics	Disponibiliza estatísticas sobre cada interface monitorizada num dado dispositivo
History	Regista e armazena todas as estatísticas periódicas de uma dada rede
Alarm	Compara variáveis com limites previamente configurados. Se determinada variável atingir um limite é despoletado um evento
Host	Contém estatísticas associadas a cada terminal da rede
Host Top N	Com base no grupo *hosts*, apresenta estatísticas ordenadas em função de um determinado objeto
Matrix	Armazena tabelas de estatísticas entre um par de terminais com base no endereço MAC
Filters	Parâmetro que define critérios de filtragem de pacotes
Packet Capture	Configuração e armazenamento dos resultados da filtragem realizada no grupo Filter
Events	Define eventos, mediante tipo e a última ocorrência. Possibilita a definição de ações como por exemplo, notificar o gestor através de envio de uma *trap*
Token Ring Extension	Contém quatro grupos para definir algumas funções adicionais de monitorização para redes Token Ring

Tabela 2 - Grupo de objetos RMON2 MIB

Grupo RMON2 MIB	Função
Protocol Directory	Apresenta os tipos de protocolos disponíveis para o processo de monitorização
Protocol Distribution	Disponibiliza a informação de tráfego por protocolo
Address mapping	Mapeamento de dados entre endereços MAC e endereços IP
Network layer host	Fornece estatísticas de tráfego de uma determinada rede
Network layer matrix	Possibilita estatísticas de tráfego de uma dada origem e destino, a nível de endereços IP
Application layer host	Demonstra estatísticas de tráfego ao nível aplicacional, de diferentes protocolos, como por exemplo HTTP, FTP, SMTP, DNS entre outros
Application layer matrix	Possibilita estatísticas de tráfego de uma dada origem e destino, ao nível aplicacional
User history collection	Mostra periodicamente objetos especificados e armazena as informações. Estes dados são especificados pelo utilizador (gestor)
Probe configuration	Parâmetros de configuração do sistema RMON, como por exemplo, data e hora
RMON conformance	Descreve os requisitos de conformidade do RMON2 MIB

2.3.4 Modelo funcional

A ITU (*International Telecommunication Union*), em conjunto com a ISO, definiu um modelo de gestão para redes de comunicação denominado TMN (*Telecomunication Management Network*). O modelo funcional TMN [24] congrega cinco funções conceptuais de gestão, nomeadas pelo acrónimo FCAPS (*Fault-management, Configuration, Accounting, Performance and Security*).

Inicialmente a funcionalidade de deteção e recuperação de falhas, revelou-se a mais importante entre os sistemas de gestão de redes, dado que este era um dos aspetos mais relevantes do ponto de vista da operação e da utilização das redes de comunicação. Com o crescimento exponencial destas infraestruturas de comunicação, os requisitos em termos de qualidade de serviço aumentaram, obrigando assim ao desenvolvimento de outras áreas funcionais.

Particularmente, o modelo funcional TMN aplica-se a todos os níveis dos modelos e paradigmas de gestão (apresentados na secção 2.3), assentando assim numa classificação de funções de gestão FCAPS :

- Gestão de falhas (*Fault*);
- Gestão de configuração (*Configuration*);
- Gestão de contabilização (*Accounting*);
- Gestão de desempenho (*Performance*);
- Gestão de segurança (*Security*).

Na Tabela 3 são apresentadas as funcionalidades genéricas do modelo de gestão FCAPS. As cinco áreas constituintes, não devem ser vistas de um modo estanque, ou seja, são apenas diretrizes que podem ser úteis para estruturar um sistema de gestão e monitorização.

Todas as áreas funcionais são importantes de igual modo, no entanto, é importante salientar que os sistemas de monitorização e alarmística focam-se essencialmente nas áreas de gestão de falhas e gestão de desempenho

Tabela 3 - Funcionalidades genéricas suportadas pelo modelo FCAPS [25]

Gestão de falhas	Gestão de configuração	Gestão de contabilização	Gestão de desempenho	Gestão segurança
Detetar falhas	Inicialização de recursos	Monitorização estatística de um serviço	Utilização e monitorização de recursos	Proteção de informação
Corrigir falhas	Configuração de serviços de rede	Suporte de diferentes modos de contabilização	Identificação de problemas detetados	Gestão de acessos
Registos de erros	Distribuição automatizada de *software ou firmware*	Disponibilidade do serviço	Análise de desempenho de dados	Monitorização de carga
Gestão de alertas	Configuração remota	Registos de estatística	Análise histórica de Registos	Análise e estatística de dados
Filtrar notificações	Descoberta automática de equipamentos	Auditorias	Gestão de tráfego	Identificação de requisitos de segurança
Testes de diagnóstico	Armazenamento e restauro de dados	Suporte a diferentes modos de contabilização	Taxa de erros ocorridos	Gestão de segurança da informação distribuída

Gestão de Falhas

É uma das áreas funcionais que se destaca pela sua importância no processo de gestão de redes de comunicação. Uma das atividades fundamentais da gestão de falhas consiste na deteção de erros, desta forma, após o processo de deteção, deverão ser despoletadas ações de diagnóstico e recuperação de erros.

O método de deteção de erros é realizado com base em ações de monitorização de eventos, mediante a ocorrência de alertas produzidos pelos dispositivos de rede, assim como, a deterioração de componentes ou desempenho da rede, ou eventualmente a falha de aplicações. Um exemplo característico de falhas que podem desencadear alertas, são as falhas de *hardware* ou a transposição de parâmetros pré-estabelecidos, que por norma desencadeiam a produção de registos de erro (*logs*).

Vulgarmente uma falha origina vários alarmes, bem como erros a ela associados. Deste modo, o diagnóstico de erros é realizado de acordo com os lapsos detetados, determinando a sua causa comum.

Por norma, a deteção ou o diagnóstico de um erro, gera uma notificação do problema, que ajuda na tomada de decisão por parte do sistema ou do gestor de rede.

Após as fases de deteção e diagnóstico, torna-se essencial instigar ações de recuperação. Um ponto importante a ter em consideração são os erros que podem surgir e sobretudo, avaliar o impacto que estes causam, podendo existir a necessidade de uma alteração na configuração de um equipamento de rede ou eventualmente a sua substituição, em caso de avaria [3].

Gestão de Configuração

O processo de gestão de configuração abrange um conjunto de funções, destacando-se o procedimento de recolha, monitorização e alteração de informação de configuração do sistema de comunicação, de modo a controlar possíveis modificações, nomeadamente aos níveis de *hardware* ou *software*.

Ao nível do serviço, a gestão de configuração possibilita transformar um serviço de rede, modificando os seus parâmetros, ou desativando o mesmo.

Paralelamente, a sincronização da informação gerida na rede com a informação de gestão, fica a cargo deste processo (gestão de configuração).

Nos sistemas de gestão mais sofisticados é frequente que, a partir da informação dos agentes existentes nos equipamentos, seja possível ter a visão topológica da rede, bem como detetar de forma automática todos os equipamentos existentes nessa mesma infraestrutura [26].

Gestão de Contabilização

A função de contabilização ganha mais relevo em redes comerciais, pois tem como principal função, definir custos de utilização dos serviços prestados.

No entanto, em redes não comerciais, a contabilização da utilização de recursos pode ser uma atividade importante a ter em conta. Através dela é possível determinar padrões de utilização de recursos por parte dos utilizadores, o que poderá auxiliar o gestor de rede na tomada de decisões.

Embora não exista uma tarifa associada por utilizador nas redes não comerciais, os equipamentos e meios de comunicação têm um custo, nomeadamente no que concerne à aquisição e manutenção dos mesmos. Tendo em conta os fatores referidos, é importante, em alguns casos, ter uma noção sobre os recursos consumidos por parte dos utilizadores, para que seja possível quantificar e justificar os custos de atualização e eventual escalabilidade da rede.

A gestão de contabilização prevê, ainda, um papel fundamental no suporte de auditorias, determinando por exemplo, a utilização concreta de recursos de um utilizador, mediante a análise de comportamentos suspeitos. Portanto, este tipo de gestão assume um requisito essencial no que diz respeito às questões legais, prevenido assim a criminalidade informática [3].

Gestão de Desempenho

Como função principal, neste modelo, destaca-se a monitorização do estado dos elementos de rede, ao nível físico e lógico. São então analisadas informações do desempenho de um determinado serviço, recolhendo e avaliando a informação em tempo real e despoletando alertas, quando os valores estão fora dos limites pré-estabelecidos. Congrega ainda funções que analisam as tendências, de modo a que seja possível prever e antecipar problemas ao nível do desempenho.

O modelo de gestão de desempenho trata igualmente da recolha e tratamento de dados relativamente ao comportamento dos objetos geridos (agentes), sendo fundamental para suportar as atividades de configuração, gestão de falhas e planeamento de rede.

Os sistemas mais elaborados poderão recorrer a informação de desempenho, de modo a estabelecer modelos comportamentais na rede, diagnosticar os problemas desta e prever o seu desempenho futuro [27].

Gestão de Segurança

Como principal finalidade, a gestão de segurança disponibiliza funções de monitorização e controlo dos mecanismos de segurança em utilização num determinado sistema. Concede múltiplos serviços de defesa aos mais diversos níveis de segurança e estabelece o controlo de acessos dos serviços, com recurso ao controlo de privacidade, confidencialidade e integridade da informação [26].

As principais atividades da gestão de segurança são a definição de utilizadores, grupos e respetivos privilégios, a identificação de requisitos de segurança associados aos diversos recursos de rede, bem como, a configuração e monitorização de sistemas de segurança, como por exemplo, *firewalls*, e o registo em *log* de incidentes pertinentes [3].

2.4 Metodologias de gestão de rede

As metodologias de gestão de rede assentam em dois pilares fundamentais: a gestão em banda (*In-Band Management*, IBM) e a gestão fora de banda (*Out-of-Band Management*, OOBM). Ambos os conceitos visam diferenciar o modo como o tráfego de gestão (*control-plane*) é transportado na rede, tendo por base as características suportadas pelos nós que a decompõe [28].

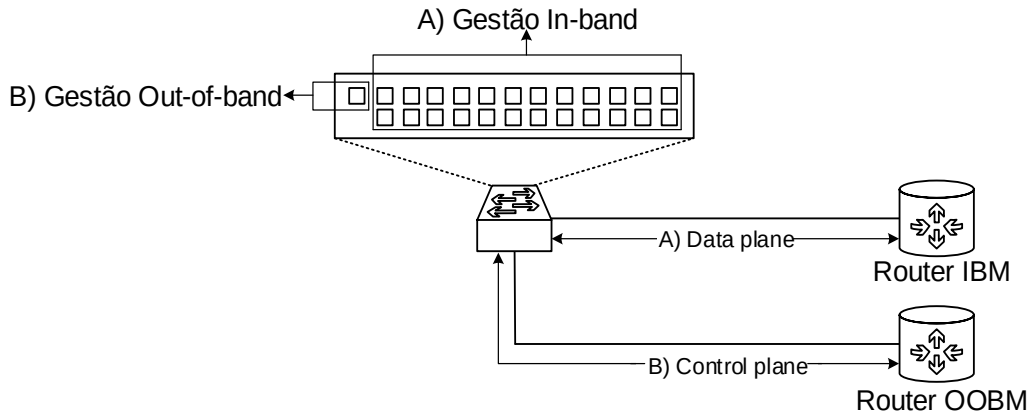


Figura 10 - Metodologia de gestão IBM versus OOBM

2.4.1 Gestão de rede *in-band*

A gestão de rede IB assegura a gestão dos equipamentos com auxílio da rede de dados, garantindo um modelo elementar no que concerne ao processo de administrar e gerir uma rede [29].

O modelo de gestão *in-band* partilha o mesmo meio físico de comunicação da rede de dados, como tal, a segmentação lógica do tráfego de gestão (*control-plane*) e do tráfego de dados (*data-plane*) é realizada com recurso a redes virtuais locais (*Virtual Local Area Network*, VLAN). Como boas práticas, poderão ser agregadas duas ou mais interfaces (*Link Aggregation Group*, LAG), a fim de criar redundância nas interfaces de acesso do equipamento, aumentando significativamente a disponibilidade de comunicação com o elemento de rede.

Embora este método não seja a solução ideal, é comumente utilizado pela carência de *hardware* dedicado por parte das instituições, nomeadamente ao nível dos componentes passivos e ativos, os quais devem ser totalmente distintos da rede de transporte de dados.

Sob compromisso de prejuízo na simplicidade desta metodologia, a dimensão elevada de dispositivos, bem como o número de serviços críticos de rede, deverá ser considerada na adoção deste modelo. Tendo em conta uma possível falha da rede de dados, a gestão de rede *in-band* deixa de ser adequada, a qual, fica comprometida pela perda de acesso aos dispositivos que apresentam anomalias, impossibilitando assim a resolução do problema. Para corrigir tal situação, torna-se extremamente importante uma ligação física alternativa que seja totalmente independente da rede de dados dos clientes (gestão *out-of-band*).

2.4.2 Gestão de rede *out-of-band*

A gestão de rede OOB proporciona um meio de comunicação diferenciado, capaz de dissociar completamente o tráfego de gestão do tráfego de dados [29]. A diferença desta metodologia prende-se, essencialmente, na separação física do tráfego de gestão, providenciando uma forma de aceder aos equipamentos, no caso de indisponibilidade da rede de transporte de dados ou congestionamento da mesma. A adoção deste modelo poderá ser vantajosa face à gestão de rede *in-band*, uma vez que possibilita um caminho de acesso direto aos nós de rede, podendo ser utilizadas em simultâneo, ambas as metodologias de gestão de rede apresentadas, a fim de criar dois meios de comunicação distintos e redundantes.

Torna-se importante salientar que a rede de gestão fora de banda, possibilita ainda uma maior segurança na rede pelos seus métodos de segregação de tráfego, sendo de referir que existem dispositivos de rede que não possuem, ou não devem ter, ligação direta à rede de dados, como por exemplo, fontes de alimentação ininterruptas (*Uninterruptible Power Supply*, UPSs), sensores de temperatura, entre outros [28].

2.5 Plataformas de gestão de rede

A gestão dos sistemas e redes de comunicação de dados exige a utilização de dispositivos e/ou plataformas de gestão especializadas, que possam cobrir e atuar nas diversas áreas funcionais estabelecidas pelo modelo FCAPS (2.3.4).

As plataformas de gestão de redes fornecem funcionalidades de gestão a diversos níveis, integrando diferentes ambientes protocolares, permitindo assim, uma melhor administração operacional das redes de grande dimensão.

Em geral, as plataformas de gestão consolidam características que as tornam abrangentes, por um lado, e flexíveis por outro. Entre essas características enumeram-se de seguida as principais:

- Dão suporte a uma variedade de protocolos e tecnologias de gestão;
- Possibilitam a integração e a monitorização de diferentes elementos de rede;
- Contemplam uma estrutura modular e distribuída, possibilitando a interação entre os módulos localizados em diferentes sistemas;
- Integram interfaces normalizadas para interações de gestão com outros sistemas;
- Suportam interfaces gráficas, permitindo assim controlar as atividades de gestão, de uma forma mais simples e eficaz;
- Incluem conjuntos básicos de aplicações, por exemplo, aplicações de configuração, desempenho, contabilização e falhas.

Nas secções posteriores serão identificadas e caracterizadas um conjunto de ferramentas *open source* de gestão e monitorização de redes, as quais foram objeto de estudo e implementação no presente projeto.

2.5.1 Prometheus

O Prometheus é uma ferramenta *open source* com o propósito de monitorizar serviços e aplicações. Esta plataforma tem por base a recolha de métricas provenientes dos sistemas e a análise de regras e expressões, podendo estas acionar alertas caso alguma condição se verifique. Uma das características principais do Prometheus é a sua arquitetura orientada ao serviço, sendo este desenvolvido para ambientes dinâmicos, em que a autodescoberta de serviços é essencial.

Por sua vez, o Prometheus engloba-se em diferentes níveis do modelo FCAPS, podendo assim, ser utilizado como ferramenta de gestão de falhas, desempenho e segurança.

Para além das funcionalidades fundamentais descritas anteriormente, é importante realçar também as seguintes características [30]:

- Suporta um modelo de dados multidimensional (*time series*);
- Possui uma linguagem própria (PromQL) para realizar *queries* em formato *time series*;
- Recolhe métricas dos equipamentos através de um modelo proprietário (*pull*) e através do protocolo HTTP;

- Permite uma vasta gama de exportadores de dados (conceito de agente): MongoDB, PostegreSQL, Elasticsearch, SNMP, Netgear, Nagios, entre outros;
- Retrocompatibilidade com múltiplas APIs (*Application Programming Interface*): GitHub, OpenWeatherMap, Docker Cloud, Mozilla Observatory, entre outros;
- Integração com diferentes ferramentas de monitorização estatística, como por exemplo Grafana, Kibana, entre outros;

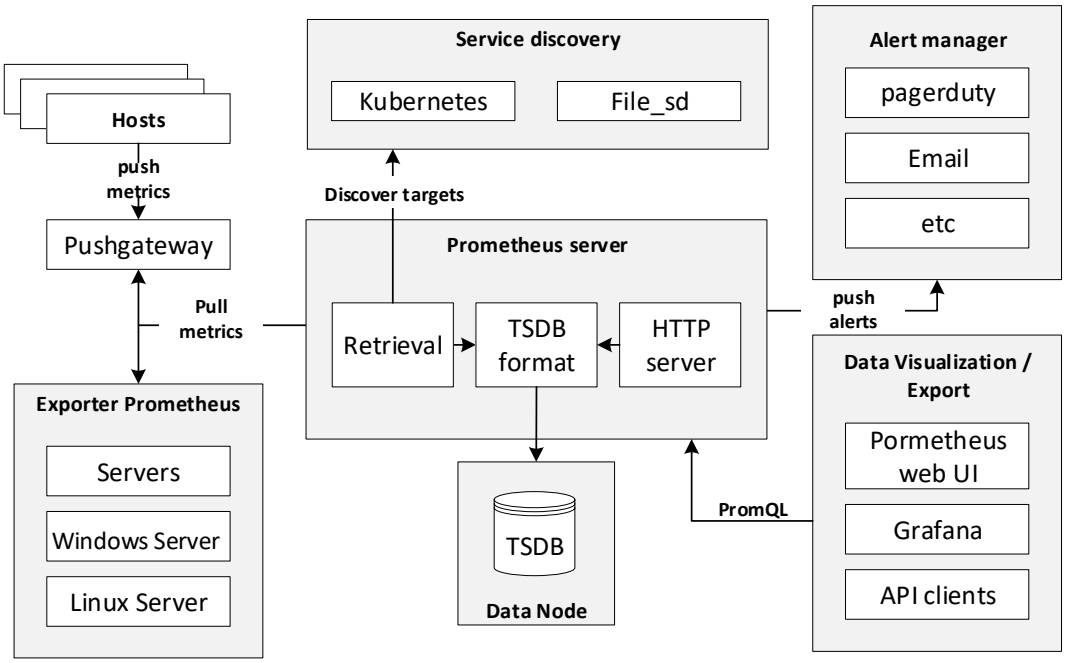


Figura 11 - Arquitetura da ferramenta Prometheus

A arquitetura da plataforma Prometheus (Figura 11) é baseada em vários componentes [30]:

- Servidor Prometheus, responsável pelo encaminhamento e armazenamento de dados (*time series*);
- Módulo de descoberta de serviços (*service discovery*).
- Módulo *Exporter*, que permite integrar bibliotecas para os *hosts* a monitorizar através de métricas;
- Componente de *proxy* (*push gateway*), sendo este de uso opcional;
- Gestor de alertas (*alert manager*);

O funcionamento desta plataforma consiste na interação entres os diferentes componentes anteriormente referidos. Através de intervalos de tempo parametrizados (`prometheus.yml`), o Prometheus *server* recolhe métricas dos *hosts*, fazendo uso do protocolo HTTP. Estas métricas

em formato *time series* são guardadas numa base de dados (*Time Series Database*, TSDB).

Adicionalmente, o Prometheus disponibiliza diversas bibliotecas em diferentes linguagens que são utilizadas para monitorizar as aplicações. O processo de monitorização de sistemas, nomeadamente servidores Linux, *routers*, entre outros, fica a cargo de um módulo *exporter*, o qual recolhe as métricas dos sistemas monitorizados e as retro compatibiliza para o formato *time series*, reconhecido pelo Prometheus *server*. O módulo *exporter* pode ser interpretado como um agente que permite monitorizar um dado nó de rede.

A descoberta de serviços fica a cargo da API Kubernetes, a qual é utilizada nativamente pelo Prometheus. Este conceito inovador para monitorização, preconiza a descoberta automática de serviços através de uma lista de instâncias que são executadas dinamicamente, sem necessitar obrigatoriamente de um endereço IP para identificar um serviço de um dado *host*. Este tipo de descoberta ocorre ao nível aplicacional.

As consultas de informação de monitorização, por parte do utilizador, são realizadas através do módulo de visualização de dados, numa linguagem de programação PromQL. Nativamente esta ferramenta disponibiliza uma interface gráfica *web*, embora a gestão desta plataforma seja mais orientada ao uso de expressões, de modo a observar os resultados em tabelas ou gráficos. Para colmatar a dificuldade de utilização das expressões, os administradores de rede poderão integrar ferramentas próprias para visualizar os dados, como por exemplo o Grafana.

2.5.2 Cacti

O Cacti é uma ferramenta *open source* cuja principal funcionalidade recai na análise da gestão de desempenho através de gráficos. A utilização de *plugins* permite um leque mais alargado no concerne às diferentes áreas funcionais da gestão (FCAPS).

Com recurso a uma interface *web*, o Cacti disponibiliza um *frontend* para o sistema de *log* de dados e elaboração de gráficos do tipo RRDtool (*Round-Robin Database*) [31]. Esta ferramenta contempla inúmeras funcionalidades, merecendo destacar a facilidade de criação e configuração de gráficos RRD com base na sua interface *web*, bem como o suporte para *scripts* e comandos externos, e a recolha de dados com base no protocolo SNMP [32].

O modelo arquitetónico preconizado pelo Cacti recorre a um *pooler* de modo a efetuar a recolha periódica dos dados aos sistemas a monitorizar (Figura 12). Depois dos dados provenientes dos sistemas serem recolhidos pelo *pooler*, são então armazenados em ficheiros RRD. Os ficheiros RRD armazenam os dados noutro ficheiro com um tamanho fixo utilizando

a metodologia *First In, First Out* (FIFO). Para agregar os dados são criados vários RRAs (*Round Robin Archives*) dentro de um único ficheiro RRD. Os RRAs representam medições, podendo estas ser diárias, semanais, mensais e anuais, ou definidas livremente pelo utilizador [33]. No processo de armazenamento das configurações inerentes ao sistema, é utilizada uma base de dados MySQL.

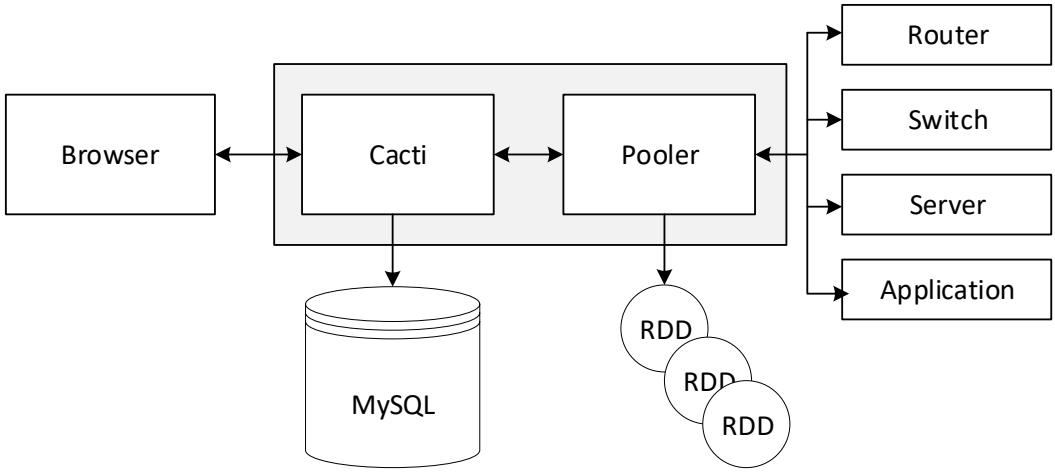


Figura 12 - Arquitetura da ferramenta Cacti

No que respeita à interface disponibilizada ao utilizador, esta providencia uma gestão simples das particularidades do sistema, através de uma aplicação *web based* desenvolvida em PHP (*Hypertext Preprocessor*) [34].

2.5.3 Zabbix

O Zabbix é uma plataforma *open source* de monitorização de sistemas que visa supervisionar duas áreas funcionais distintas, nomeadamente a vertente de gestão de falhas e na gestão de desempenho. A sua arquitetura base é distribuída, assentando num sistema central de monitorização, num conjunto de *proxies* para monitorização distribuída e nos clientes (nós).

Como principais características do Zabbix, enumera-se as seguintes [35]:

- Descoberta "automática" de dispositivos de rede e servidores;
- Monitorização distribuída, com base em *proxies*, sendo a informação recolhida e disponibilizada num sistema central;
- Suporte de mecanismos de autenticação de utilizadores;
- Facilidade de integração com diferentes sistemas, através de APIs desenvolvidas em diversas linguagens de programação;
- Processo de monitorização com recurso a diferentes protocolos SNMP (v1, v2 e v3), IPMI (*Intelligent Platform Management Interface*), JMX (*Java Management Extensions*), OBDC (*Open Database Connectivity*), SSH, Telnet, HTTP(S), TCP/UDP;
- O modelo gestor-cliente (Zabbix *server* e Zabbix *agent*) dá suporte a uma grande variedade de sistemas operativos baseados em Unix e Windows.

Ao nível arquitetónico, o Zabbix engloba três entidades essenciais, entre os quais, um servidor *web*, um servidor Zabbix e um servidor de base de dados (Figura 13). Por outro lado, não sendo de uso obrigatório o Zabbix-*proxy* poderá ser utilizado para balanceamento de carga.

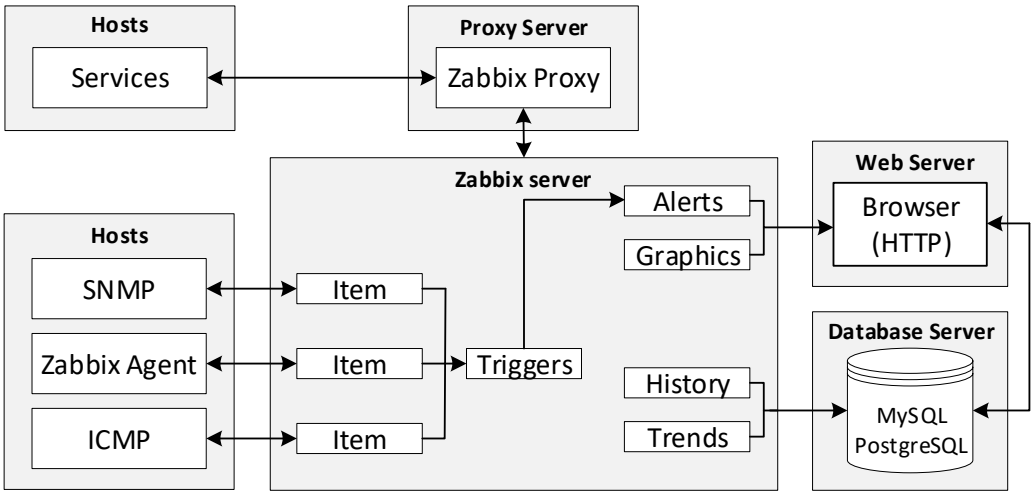


Figura 13 - Arquitetura da ferramenta Zabbix

Independentemente dos componentes do Zabbix poderem estar distribuídos, a sua configuração é centralizada, diminuindo assim, a probabilidade de configurações inconsistentes entre componentes [36].

Embora a topologia arquitetónica comummente utilizada não recorra à segmentação de componentes, o funcionamento tradicional do Zabbix poderá ser suficiente para monitorizar uma rede de dados. Contudo, o Zabbix torna-se uma ferramenta escalável, poderosa e diferenciada, por poder ser personalizável às características dimensionais de uma rede. Em casos de redes de tamanho elevado, o Zabbix permite segregar os seus componentes constituintes em *hardware* dedicado e distinto (Figura 13), sendo possível ainda, a criação de *clusters* de base de dados, a implementação de *clusters* de *proxies* e *clusters* de servidores Zabbix, com o objetivo de melhorar o seu desempenho garantindo o conceito de *cluster* de alta disponibilidade (*High-Availability cluster*, HA *cluster*) [36].

O funcionamento do Zabbix suporta vários tipos de monitorização. Os dados de monitorização de um *host* são representados por *Items* e este pode ser obtido de diferentes formas. Como metodologia de controlo para os diferentes *Items* o Zabbix recorre a diferentes tipos de monitorização [37].

O conceito de *simple checks* verifica a disponibilidade de múltiplos serviços (SMTP, HTTP, FTP, ICMP, entre outros) [37]. Para utilizar o tipo de monitorização *simple check* em servidores, é necessário configurar o *software Zabbix Agent*, o qual possibilita uma monitorização ao nível local, sendo responsável também pelo envio dos dados obtidos ao *Zabbix Server*.

Por outro lado, para monitorizar equipamentos como *switches*, *routers*, entre outros, são utilizados SNMP *checks*, sendo importante destacar que este tipo de dispositivos, por norma, suporta o protocolo SNMP, e não são passíveis de instalar e configurar o agente Zabbix [37].

Toda a informação de gestão e configuração é armazenada numa base de dados relacional. O suporte de soluções de comunicação normalizadas é feito através do protocolo SNMP, possibilitando operações de *polling* e *trap* (notificações). Adicionalmente o Zabbix possibilita o armazenamento dos dados em dois formatos distintos: *History* e *Trends*. Por sua vez, o mecanismo *History* guarda cada valor recolhido, enquanto o mecanismo *Trends* recolhe uma média dos valores em base horária, consumindo assim menos recursos computacionais.

O mecanismo de notificação é extremamente flexível, possibilitando que praticamente qualquer evento seja despoletado por um *trigger*, gerando assim, uma mensagem de alerta que poderá ser enviada por *e-mail* (entre outros).

A produção de relatórios e a visualização de dados são também bastante importantes, permitindo a análise de valores numéricos ou gráficos de vários tipos. Todos os relatórios, estatísticas e parâmetros de configuração, são acessíveis através de um *front-end* gráfico *web-based* [38].

2.5.4 Nagios

O Nagios é uma ferramenta de monitorização de sistemas e redes que se enquadra na área funcional da gestão de falhas. Muita da sua versatilidade e potencial advém da sua arquitetura extremamente simples, centrada num *daemon* que se encarrega do agrupamento de diferentes tarefas de monitorização e de notificação de eventos.

Como ferramenta que se destaca no processo de gerir falhas, a sua principal funcionalidade passa por analisar o estado de operabilidade dos *hosts* e dos serviços discriminados, alertando o gestor de rede quando determinadas situações anómalas se confirmem. Supondo que um *host* fique indisponível por parte da rede, o Nagios despoleta, por exemplo, um *e-mail* de modo a notificar o utilizador da falha ocorrida. Embora a sua página *web* não permita fazer todo o tipo de configurações, como por exemplo, adicionar *hosts* e/ou serviços a monitorizar, permite, ainda assim, apresentar o estado de todos os dispositivos que estão a ser monitorizados [39].

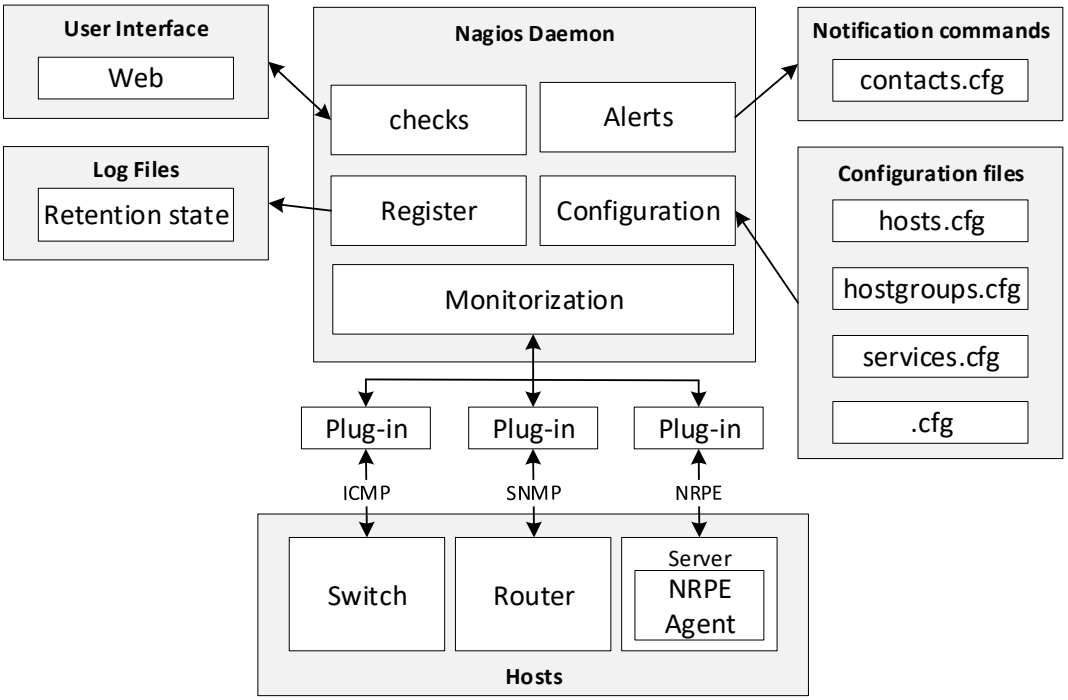


Figura 14 - Arquitetura da ferramenta Nagios

Utilizando quer os *plug-ins* disponíveis para a distribuição base quer *plug-ins* adicionais, o Nagios executa a monitorização dos serviços, de acordo com a informação do ficheiro de configuração. Os *plug- ins* oficiais permitem monitorizar os serviços mais comuns, tais como, ICMP, SNMP, HTTP, já referidos anteriormente, mas também IMAP (*Internet Message Access Protocol*), SSH (*Secure Shell*), POP3 (*Post Office Protocol*), DHCP (*Dynamic Host Configuration Protocol*), entre outros [40]. Adicionalmente importa, destacar o agente NRPE (*Nagios Remote Plugin Executor*) habitualmente utilizado para monitorização de servidores.

O agente NRPE permite a monitorização de serviços, nomeadamente: a utilização de CPU (*Central Process Unit*), a utilização de disco, a utilização de memória, entre outros. Os *plug-ins* são processos executáveis compilados ou *scripts* (*scripts* Perl, etc.) que podem ser escritos em múltiplas linguagens de programação. Nativamente a comunidade do Nagios dispõe de imensos *plug-ins*, o que providencia a monitorização de um elevado número de serviços e *hosts*, destacando assim a sua massificação e adesão por parte dos administradores de rede [40].

Ao nível das ações de monitorização despoletadas pelo Nagios, é importante referir, que os resultados destes testes (*checks*) são armazenados em ficheiros de *log*. Por sua vez, o estado dos equipamentos e serviços é guardado na base de dados de retenção de estados (`retention.dat`).

As ações de monitorização iniciadas pelo *daemon* do Nagios são designadas por *active-checks*, já as ações de monitorização iniciadas por processos externos e enviadas ao Nagios são designadas por *passive-checks* [39].

Cada *check* proveniente de um *plugin* pode retornar diferentes estados, nomeadamente: *OK*, *WARNING*, *CRITICAL* e *UNKNOWN*. Por definição são enviados ao *host*, três *checks* de verificação de estado, após a falha no primeiro *check*. Caso a resposta não seja retribuída, ou seja diferente do estado *OK*, é enviado um alerta ao utilizador [39].

De um modo geral, a monitorização de um *host* é realizada através de ficheiros de configuração. Por norma, a configuração para a monitorização de um *host* pode ser realizada apenas nos seguintes ficheiros: `nagios.cfg`, `resource.cfg`, `cgi.cfg`. Como ficheiro central onde é definido diversas parametrizações do Nagios, enumeram-se os *Object Defination Files*.

Na Tabela 4 é descrito o papel desempenhado por cada um dos ficheiros supracitados, os quais têm um papel preponderante, no que concerne à gestão operacional da ferramenta Nagios [39] [40].

Tabela 4 - Função dos principais ficheiros do Nagios

Ficheiros	Função
`nagios.cfg`	Ficheiro de arranque do *deamon* Nagios; Indica os restantes ficheiros a consultar. Deste modo, este ficheiro necessita das localizações dos diferentes ficheiros de configuração.
`resource.cfg`	Guarda as macros criadas pelo utilizador; A principal função deste ficheiro é esconder informações sensíveis, tais como *passwords* CGIs (*Common Gateway Interface*).
Object Defination Files	Define a localização nativa dos *hosts*, serviços, *host groups*, *contacts*, *commands*, entre outros. Descreve todos os ficheiros, os quais, o administrador de rede deverá parametrizar para recorrer ao processo de monitorização e alarmística do Nagios.
`cgi.cfg`	Define a parametrização e customização das ações da interface *web*.

2.5.5 Zenoss Core

O Zenoss Core tem como o principal objetivo monitorizar o estado dos nós de rede oferecendo uma opção fiável para gerir e administrar uma infraestrutura de sistemas informáticos. Analisando o seu perfil funcional FCAPS, podemos classificar que esta plataforma cobre primordialmente a gestão de falhas, embora também providencie tarefas de gestão de desempenho graças à sua integração com o RRDTool.

O modo de funcionamento do Zenoss Core tem por base uma arquitetura bastante peculiar. Todos os elementos podem ser parametrizados através de linha de comandos no entanto, esta ferramenta contempla uma interface *web* que oferece um ambiente gráfico dinâmico, fornecendo ao utilizador uma navegação interativa, viabilizando a gestão automatizada dos dispositivos (*discovery*) ao nível do seu estado (*avaliability*) e desempenho (*performance*), bem como proporciona ainda, a gestão de eventos (*events*) e relatórios de sistema, entre muitas outras funcionalidades.

A sua arquitetura modelar é dividida em 4 camadas fundamentais (Figura 15): camada de utilizador; camada de dados; camada de processamento e camada de recolha [41].

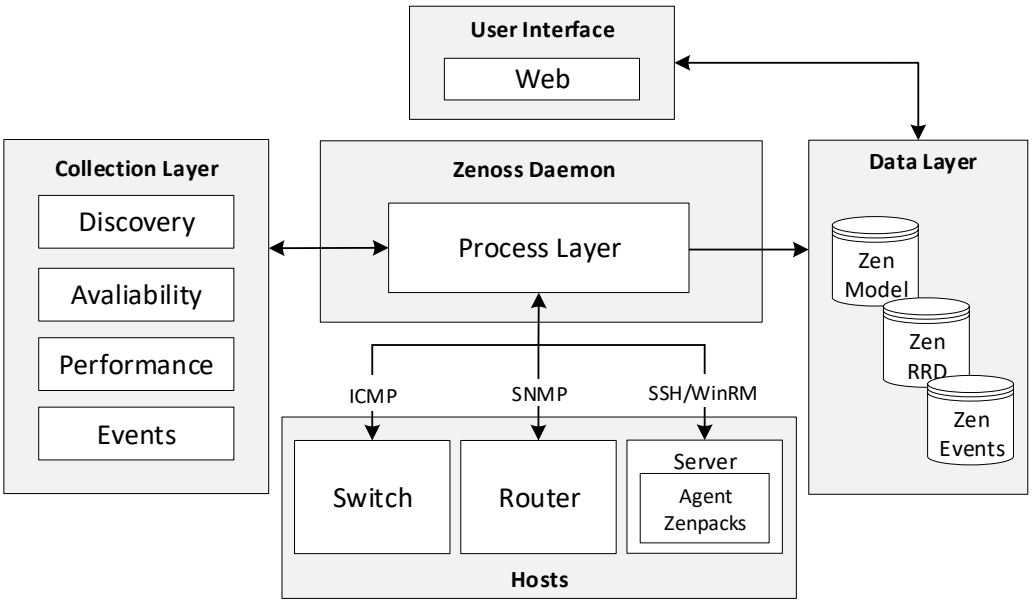


Figura 15 - Arquitetura da ferramenta Zenoss

Adicionalmente a interface *web* é estruturada com base na *framework* Zope utilizando Pyton como linguagem de programação orientada a objetos, dando ainda suporte a diferentes tecnologias, nomeadamente JavaScript, Mochi Kit, ExtJS, YUI, entre outros [41].

O tratamento dos dados é realizado em três tipos distintos de base de dados: ZenModel; ZenRRD; ZenEvents. A informação inerente aos serviços dos *hosts* é monitorizada a partir da camada de recolha (*collection layer*). Posteriormente, os dados são encaminhados para a camada de processamento (*process layer*) que separa a informação e trata de armazenar a mesma na respetiva base de dados [41].

Para o armazenamento de dados provenientes de configurações do Zenoss Core, é utilizado o paradigma de base de dados CMDB (*Configuration Management Database*), associado ao conceito ZenModel.

Os eventos, como por exemplo *logs* ou notificações, são arquivadas numa base de dados MySQL (ZenEvents). Estes eventos, podem gerar ações, como envio de *e-mail* ou SMS (*Short Message Service*), caso se verifique uma situação de falha.

Por sua vez, os dados provenientes da monitorização estatística são recolhidos e armazenados em ficheiros RRD através do RRDTool integrado no Zenoss Core.

Do ponto de vista lógico, podemos considerar o RRD como uma terceira base de dados para guardar a informação de desempenho dos *hosts*, como por exemplo, a utilização de CPU, disco, memória ou tráfego de entrada e saída, entre outros.

As tecnologias suportadas por esta plataforma são um pouco distintas das ferramentas de monitorização alarmística tradicionais. Destaca-se o protocolo SNMP como sendo o mais genérico para monitorizar equipamentos centrais de comunicação (*switches*, *routers*, etc.) que não permitem a instalação de agentes.

No que concerne à metodologia utilizada para monitorizar servidores Linux (*zenpack*s) e servidores Windows (*Windows Remote Management*, WinRM), o Zenoss Core distingue-se por disponibilizar pacotes específicos baseados em tecnologias SSH e WMI (*Windows Management Instrumentation*) respetivamente [42] [43]. A forma de monitorizar um servidor pelo protocolo SSH, não é habitual numa ferramenta de gestão de falhas, podendo oferecer uma mais-valia, aliando assim, a gestão de segurança que o protocolo SSH propõe.

2.5.6 Syslog-NG

O Syslog-NG destaca-se pela sua flexibilidade, alta escalabilidade e fornecimento de uma solução centralizada no que respeita ao processo de gestão de *logs* [44]. Analisando o perfil funcional desta ferramenta, podemos aferir que a mesma se enquadra no modelo de gestão de segurança.

A principal função do Syslog-NG é analisar as mensagens recebidas de ficheiros, encaminhando estas para os destinos especificados (fontes). Estas mensagens são originadas *por terminais* diferentes origens. Cada fonte define a origem das mensagens rececionadas pelo Syslog-NG.

Para possibilitar que objetos distintos (fontes e destinos) sejam uniformizados em um ou mais caminhos, são utilizados os *logs paths*. Cada mensagem rececionada de uma dada origem é então enviada para os destinos listados na instrução de *log* (*log statement*) [45]. Ainda assim, os caminhos de *log* poderão conter filtros, os quais visam selecionar o envio de mensagens de determinada fonte (exemplo: aplicação), mediante a seleção e a validação das regras dos filtros para um dado destino.

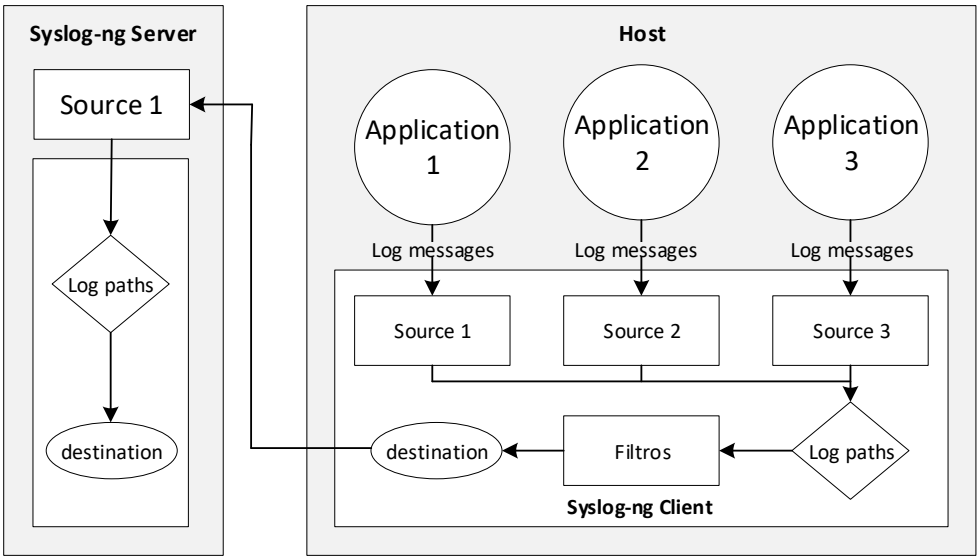


Figura 16 - Arquitetura e *workflow* do Syslog-ng [28]

Como pode ser verificado na Figura 16, a arquitetura e funcionamento genérico do Syslog-NG, retrata a metodologia supramencionada. Concretamente, ao analisarmos um exemplo de funcionamento do módulo do cliente Syslog-NG, comprovamos a receção das mensagens de *log* de diferentes aplicações através de várias origens, sendo estas mensagens encaminhadas pelos *logs paths* para um destino, sob pena da análise prévia das regras dos filtros.

Neste caso, as mensagens de *log* cumprem as regras definidas pelos filtros e são expedidas para o seu destino (servidor Syslog-NG). Após a receção das mensagens de *log* pelo servidor Syslog-NG, este analisa a sua lista de *log path*, de modo a encaminhar a mensagem para o devido destino.

2.5.7 Graylog

O Graylog permite ordenar e classificar os *logs*, disponibilizando assim, uma gestão de *logs* totalmente integrada, baseada na indexação e análise de dados (estruturados e não estruturados) de diversos componentes ativos de rede. Esta aplicação propõe ainda uma metodologia centralizada num sistema único, facilitando assim, o controlo e/ou identificação de um possível problema [46].

Tendo em conta o aumento significativo da auditoria e segurança que esta ferramenta proporciona, a mesma enquadra-se no nível de gestão de segurança assignado ao modelo funcional FCAPS.

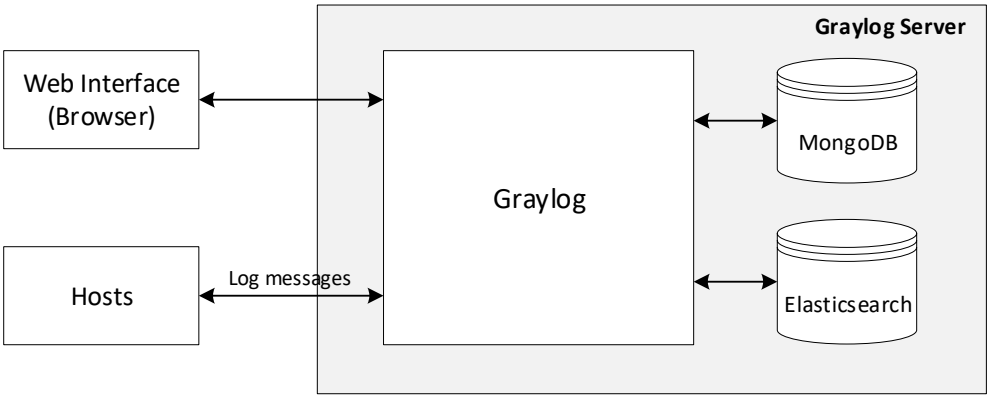


Figura 17 - Arquitetura da ferramenta Graylog

Na Figura 17 fica percetível a arquitetura proposta pelo Graylog, sendo esta subdividida em três componentes principais: a base de dados MongoDB, a qual trata do armazenamento das configurações da aplicação; a base de dados baseada em Elasticsearch, utilizada para guardar as mensagens de *log* recebidas; e o próprio Graylog, o qual é responsável pelo processo de recolha, filtragem e análise das mensagens de *log* [47]. De modo a providenciar aos seus utilizadores uma gestão de tarefas mais eficaz, o Graylog disponibiliza ainda, uma interface *web* que proporciona, de forma gráfica, a análise e processamento de *logs*.

Tabela 5 - Conceitos-chave do funcionamento do Graylog

Conceitos	**Funcionalidades**
Inputs	Definem o método de receção de mensagens; Por definição são suportados os seguintes métodos de recolha de *logs*: Syslog, GELF (*Graylog Extended Log Format*), Beats/Logstash, CEF (*Common Event Format*), JSON, Netflow (UDP), Plain/Raw Text; Recolha de *logs* via camada de transporte (TCP/UDP), ou métodos da camada aplicacional AMQP (*Advanced Message Queuing Protocol*), Kafka.
Outputs	Constituem métodos de encaminhamento de dados para os *hosts* a monitorizar; Por omissão, são suportados os *outputs* STDOUT (*Standard Output*) e GELF.
Pesquisas	Possuem uma interface *web*; Utilizam uma sintaxe elementar e disponibilizam intervalos de tempo relativos ou absolutos.
Streams	Promovem uma forma de selecionar as mensagens de *log* recebidas, encaminhando e categorizando as mesmas; Permitem alterar ou modificar as mensagens.
Índices	Unidade que determina as instâncias armazenadas no Elasticsearch; Providenciam políticas de retenção e rotação de dados.
Pipelines	Aplicam regras, ou um conjunto de políticas, a um evento específico; Definem funcionalidades, no processo de examinar, alterar, converter e excluir mensagens.

Na Tabela 5, enumeram-se os principais conceitos da arquitetura do funcionamento do Graylog [47].

Torna-se importante salientar que o Graylog é adaptativo à dimensão de uma rede, a qual, poderá contemplar um elevado número de nós, necessitando assim, de uma solução que garanta a resiliência e a alta disponibilidade [48].

A arquitetura genérica do Graylog (Figura 17) poderá ser readaptada, implementando o conceito de *cluster* aliado aos componentes de Elasticsearch e Graylog (agrega um MongoDB por cada nó de Graylog).

Poderá ainda, ser utilizado um balanceador de carga para possibilitar ter dois ou mais nós de Graylog, por forma a garantir um melhor desempenho, bem como, um aumento significativo do processamento das mensagens de *log*. O balanceador de carga propõe um método de validação dos nós de Graylog disponíveis no *cluster*, API REST (*Representational state transfer*), através da execução do protocolo ICMP via HTTP. Em caso de uma possível falha, o nó que apresentar problemas de conetividade será removido do *cluster*[48].

2.5.8 RANCID

O RANCID (*Really Awesome New Cisco confIg Differ*) é uma ferramenta que monitoriza as alterações dos ficheiros de configuração dos equipamentos de uma rede [49]. Trata-se de uma plataforma com um perfil funcional associado ao modelo de gestão de configurações preconizado pelo FCAPS.

Em termos de suporte de dispositivos de rede, o RANCID é compatível com uma grande variedade de fabricantes, tais como Cisco, HP Procurve, Juniper, Redback, Dell Force10, Alteon entre outros [49].

Como principal objetivo, o RANCID propõe a automatização no processo de recolha e armazenamento de configurações, providenciando métodos de *backup* que disponibilizam mecanismos de restauro baseados em versões mais antigas, os quais, promovem a possibilidade de analisar quando uma configuração específica foi concretizada.

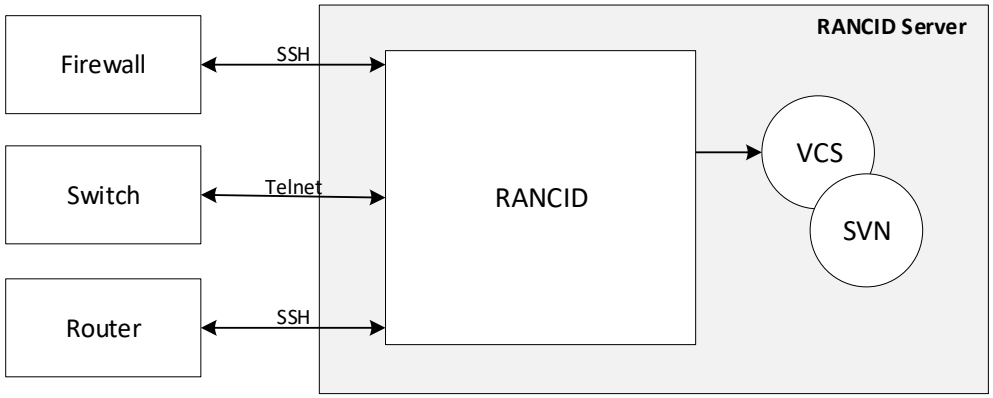


Figura 18 - Arquitetura da ferramenta RANCID [28]

A arquitetura de funcionamento do RANCID (Figura 18) centra-se no seu objetivo anteriormente referido, assim sendo, este recorre a um sistema de controlo de versões (*Version Control System*, VCS), o qual viabiliza o armazenamento de configurações através de CVS (*Concurrent Versions System*) ou o SVN (*SubVersionN*) [49].

O processo de configuração e parametrização do RANCID é realizado no ficheiro `router.db`. Este ficheiro contempla os endereços IP dos dispositivos, onde se pretende recolher as configurações e o modelo de cada dispositivo.

Para que o RANCID inicie uma ligação com o dispositivo de rede, este recorre a um protocolo de acesso remoto e à respetiva palavra-chave do equipamento. Esta conexão aos elementos de rede é efetuada através dos protocolos SSH ou Telnet.

A partir da definição do modelo de cada equipamento de rede no ficheiro `router.db`, o RANCID contém uma lista de comandos para poder extrair a configuração dos elementos de rede associados (exemplo de comando cisco: *show runnnig-config*).

Através da utilização de filtros, o RANCID poderá recolher a informação de configuração com os comandos emitidos, estando esta formatada de acordo com as regras definidas pelo utilizador. A apresentação da formatação pode conter diferentes aspetos, como por exemplo, comentários adicionais ou a omissão de determinados comandos.

Com base no ficheiro VCS, o utilizador poderá ser notificado pela receção de um *e-mail*, mediante uma possível alteração nos ficheiros de configuração atuais, com base na comparação das últimas versões armazenadas.

2.5.9 Oxidized

O Oxidized é uma plataforma que permite a visualização de configurações dos dispositivos de rede, bem como, possibilita correlacionar diferentes configurações, processando-as e armazenando-as num sistema de controlo de versões (*backups*). Assim como o RANCID, o Oxidized insere-se no paradigma de gestão de configurações proposto pelo modelo FCAPS.

Ao contrário das ferramentas mais tradicionais, como o RANCID, o Oxidized recorre a tecnologias mais modernas, utilizando novos paradigmas para o controlo de versões baseados em GIT, e novas linguagens de programação, como o Ruby, ao invés de linguagens mais antigas tais como Perl, TCL (*Tool Command Language*), entre outras [50].

O modo de execução do Oxidized distingue-se pela sua forma de automatizar o armazenamento de dados, recorrendo à instanciação de múltiplas *threads* para cada processo de recolha de configuração [50].

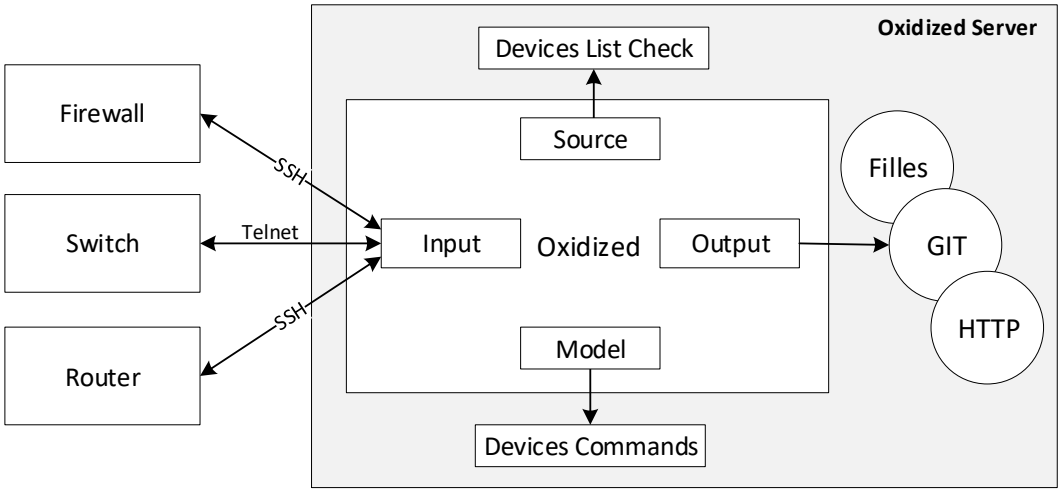


Figura 19 - Arquitetura da ferramenta Oxidized [28]

O conceito de API do Oxidized utiliza os novos modelos arquitetónicos baseados em API REST, a qual recorre a novos métodos HTTP. A API REST providencia de igual modo, a integração do Oxidized com outros sistemas para possibilitar o envio de alertas, quando existe uma alteração na configuração de um determinado dispositivo. Poderá ser integrado o Oxidized com o NMS, para despoletar o envio de notificações caso a recolha de uma configuração falhe [51].

Tabela 6 - Conceitos-chave do funcionamento do Oxidized [65]

Componentes	Funcionalidades
Source	Especifica uma lista de dispositivos para recolha de ficheiros de configuração, com base em endereço IP e *password*, entre outros; Possibilita a integração com base de dados SQL, ficheiros de texto (router.db) e HTTP.
Input	Possibilita a interação com os elementos de rede; A conexão é baseada nos protocolos de acesso remoto: SSH e Telnet.
Output	Promove métodos de armazenamento de ficheiros de configuração, com recursos a tecnologias GIT, GIT-Crypt e HTTP.
Model	Estabelece a informação a recolher dos dispositivos com base no seu modelo e nos respetivos comandos da CLI (*Command-Line Interface*); Suporta diferentes fabricantes de *hardware*: Cisco, Juniper, HP Procurve, Nokia, entre outros.

A arquitetura do Oxidized é definida por quatro componentes essenciais (Figura 19), os quais descrevem o funcionamento desta ferramenta (Tabela 6).

Capítulo 3

3 Infraestrutura de gestão da rede da Universidade do Porto

3.1 Introdução

Uma universidade é, atualmente, bem mais que um estabelecimento de ensino onde se lecionam conteúdos diversificados, que culminam para os seus estudantes na obtenção de um grau académico.

Podemos então, considerar o meio universitário, como uma organização bastante complexa, com diversos agentes, que para além da educação científica, promove uma educação empresarial, através da incubação, da investigação e promoção da inovação científica.

Um dos principais objetivos da Universidade do Porto passa por gerir estes desafios, conjugando diferentes necessidades de várias Unidades Orgânicas, sendo esta, uma tarefa complexa e que envolve múltiplos recursos humanos, materiais e financeiros.

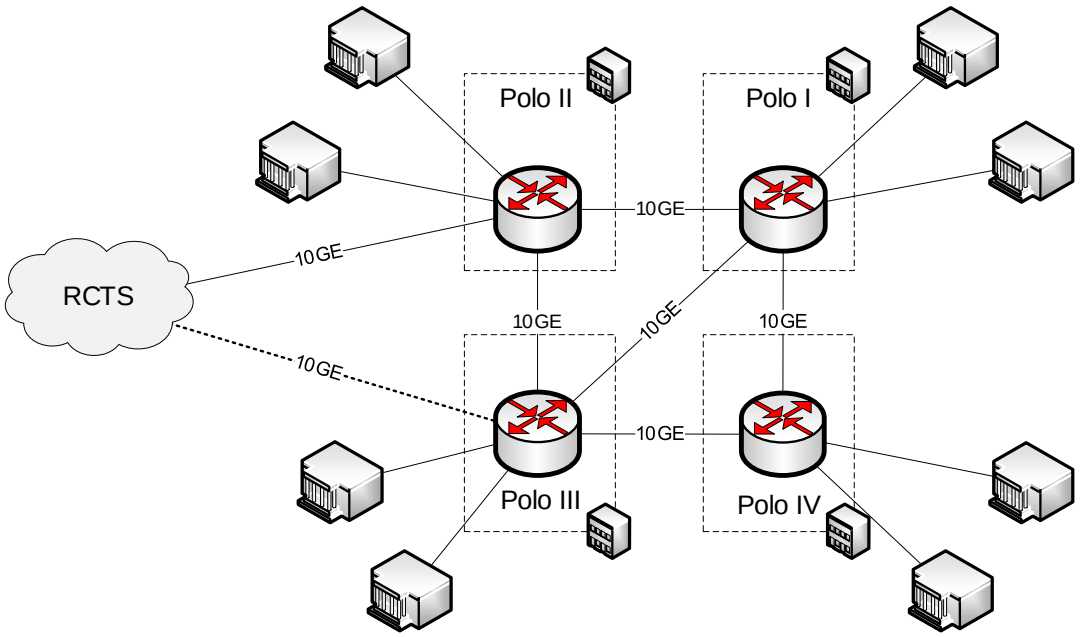


Figura 20 - Subsistema de núcleo e distribuição da Universidade do Porto

De modo a gerir estes e outros desafios, nomeadamente ao nível da gestão de infraestruturas e serviços de tecnologias de informação e comunicação (TIC), bem como no âmbito do

desenvolvimento e a utilização de serviços inovadores, a Universidade do Porto Digital (UPdigital) é, atualmente, a entidade organizacional que disponibiliza e assegura todos estes componentes, com base em unidades orgânicas que segmentam a administração da Universidade do Porto.

Devido à dimensão da rede da U. Porto, o processo de gestão e administração da rede, é realizado por equipas distintas. Neste sentido a unidade de Serviços de Redes Centrais e *Datacenters* (SRC), tem como função principal, gerir toda a rede de núcleo e distribuição da Universidade do Porto, e a unidade de Serviços de Rede Locais, tem como principal atividade, administrar a rede de acesso (Instituições e Faculdades) da Universidade do Porto.

A rede metropolitana da U. Porto fornece múltiplos serviços de conetividade entre as diferentes unidades Orgânicas, recorrendo assim a diferentes protocolos (IP, OSPF e BGP) e tecnologias (MPLS e VPLS).

Tabela 7 - Entidades do subsistema de acesso

Pólo I (Reitoria)		**Pólo III (FCUP)**	
Faculdades	FBAUP	Faculdades	FCUP
Organizações/Instituições	Reitoria		FFUP/ICBAS
	ISPUP		FLUP
	ICBAS (edifício antigo)		FAUP
	CDUP	Organizações/Instituições	CEMUP
	FIMS		PBS/EGP
	Porto Digital		CAUP
	J.Falcão		UPTEC
Residências	J.Sousa		CDUP
	Bandeirinha		CUP
	A.Cunha		LIACC
Pólo II (FEUP)			Casa Anderson
Faculdades	FEUP		Porto Digital
	FMUP	Residências	Ciências
	FCNAUP		Amaral
	FADEUP		RUCA
	FPCEUP		
	FMDUP	**Pólo IV (FDUP)**	
	FEP	Faculdades	FDUP
Organizações/Instituições	INESC		FLUP
	INEGI	Organizações/Instituições	CIIMAR (edifício antigo)
	CIPES		Coronel Pacheco
	UPTEC		ICETA
	CIIMAR		SASUP
	Porto Digital		OUP
	Vairão-ICBAS		UPTEC
Residências	Paranhos		

Como base de interligação entre as diferentes Instituições, Faculdades, Residências e outros Organismos (Tabela 7), são utilizados quatro comutadores IP localizados geograficamente em diferentes pontos de presença (Figura 20).

A ligação à *Internet* é facultada pela RCTS, operada pela FCT-FCCN (Fundação para a Ciência e a Tecnologia - Fundação para Computação Científica Nacional).

O acesso à rede da RCTS é efetuado preferencialmente pela interligação do comutador IP situado no ponto de presença do Pólo 2 (FEUP, Faculdade de Engenharia da Universidade do Porto), existindo também, uma ligação redundante, disponibilizada pelo comutador IP localizado no ponto de presença do Pólo 3 (FCUP, Faculdade de Ciências da Universidade do Porto).

Perante este cenário, podemos constatar a enorme dificuldade que pode ser gerir e administrar uma rede desta dimensão, a qual, está constantemente a evoluir, tanto na vertente física (equipamentos), quanto na vertente lógica (configurações).

Em concreto, no presente capítulo, existe o propósito de avaliar a infraestrutura de rede existente, do ponto de vista da sua arquitetura física e lógica. Ganha relevo de igual modo, analisar a rede de gestão, demonstrando o seu funcionamento, bem como a gestão operacional necessária para disponibilizar a manutenção diária da rede. Para este procedimento, torna-se vital recorrer ao modelo FCAPS, o qual irá permitir uma análise mais detalhada da rede de gestão, segmentando assim, toda a infraestrutura atual.

3.2 Análise da rede de comunicação de dados

Nesta secção serão descritos os equipamentos passivos e ativos que decompõe a topologia de rede física da Universidade do Porto, como também, todas as tecnologias e protocolos que que fundamentam a topologia lógica.

3.2.1 Arquitetura física

No que concerne à infraestrutura física da U. Porto, existem múltiplas interligações entre os seus nós constituintes. Toda a rede de núcleo e distribuição é dotada de equipamentos de rede ativos, que recorrem a uma infraestrutura passiva (*layer* 1 do modelo TCP/IP), de modo a disponibilizar diversos serviços de conectividade (Figura 21).

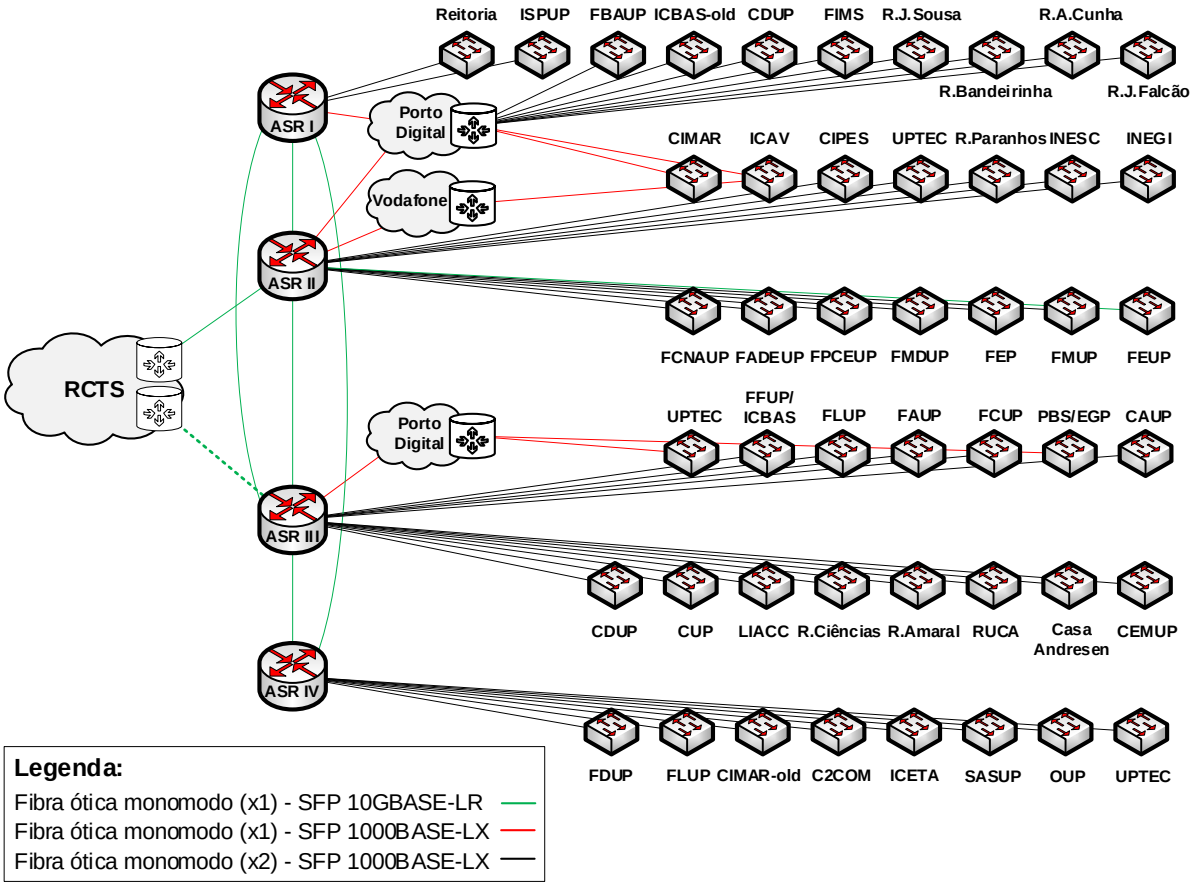


Figura 21 - Arquitetura física de núcleo e distribuição da U. Porto

A infraestrutura passiva da rede de núcleo é constituída por *transceivers* de fibra ótica (*Small Form-factor Pluggable*, SFP), que estão inseridos nas interfaces genéricas (*Network Interface Card*, NIC) dos comutadores IP (*Aggregation Services Routers*, ASR). Estes *transceivers* (10GBASE-LR) utilizam fibra monomodo, e permitem um débito até 10 Gigabits/s, com um alcance máximo de 10 quilómetros.

Na componente passiva da rede de distribuição, são empregues de igual modo, ligações em fibra monomodo que, com recurso a *transceivers* SFP (1000BASE-LX), disponibilizam débitos de transmissão até 1 *Gigabits/s*, com uma distância máxima de 10 quilómetros.

Ainda ao nível passivo, torna-se relevante evidenciar a forma como a redundância é realizada. Tendo em conta que os quatro nós que decompõem o subsistema de núcleo se encontram interligados por um anel de fibra redundante, enumera-se também a existência de duas fibras por cada interligação entre os comutadores IP (*routers*) e os comutadores *Ethernet* (*switches*) de periferia, os quais, delimitam a rede de distribuição/acesso. Isto possibilita a cada unidade organizacional atingir débitos até 2 *Gigabits/s*, através da agregação de duas interfaces físicas, salvo exceções pontuais, onde poderão existir débitos inferiores ou superiores.

Em termos geográficos, a rede da U. Porto, estende-se a locais que não possuem infraestrutura passiva para prestar serviços de conetividade. Contudo, mediante acordos pré-estabelecidos, existem dois operadores de telecomunicações que prontificam a sua infraestrutura de rede metropolitana, mais concretamente, a Associação Porto Digital e a Vodafone.

Na vertente da infraestrutura ativa, a rede da U. Porto, recorre a diferentes equipamentos, tais como, *routers*, *switches*, *firewalls*, servidores, entre outros (Tabela 8). Existem diversos servidores (centrais e locais) que são geridos por diferentes unidades orgânicas da UPDigital (Figura 22).

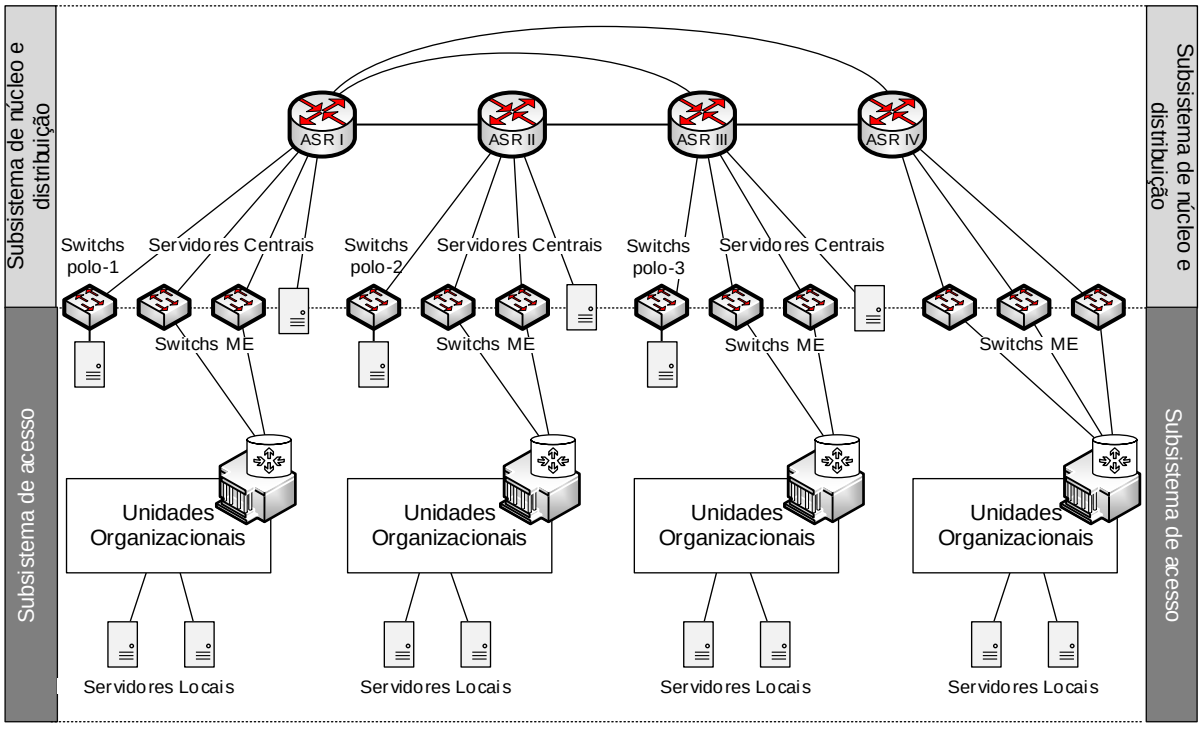


Figura 22 - Arquitetura de rede da U. Porto

Neste contexto, há servidores interligados diretamente ao subsistema de núcleo, devido à necessidade de serviços ubíquos a todas as Unidades Orgânicas da U. Porto (UOs). Entre estes, destacam-se os servidores centrais, os quais têm como principais funções: a atribuição de endereços IP de forma dinâmica (*Dynamic Host Configuration Protocol*, DHCP); o registo de eventos relevantes (*Logs*); a tradução de nomes de domínios em endereços IP e vice-versa (*Domain Name Server*, DNS); a monitorização alarmística e estatística de todos os equipamentos ativos e, a gestão *wireless*, de modo centralizado, de todos os processos de autenticação, autorização e contabilização (RADIUS).

Em relação aos equipamentos ativos, a rede da U. Porto aglomera inúmeros equipamentos, no entanto, para o presente projeto, torna-se importante descrever os principais equipamentos que decompõem a rede de gestão, conforme é representado na Tabela 8.

Tabela 8 - Principais equipamentos ativos de núcleo e distribuição da U. Porto

Equipamentos	Modelo/Descrição
Routers	Cisco ASR-9010
	Cisco ASR-9006
Switches	Cisco 2950
	Cisco 2960
	Cisco 3560
	Cisco 4500
	Cisco 3600x ME
	Cisco Nexus 5548
	Cisco Nexus 2248 FEX
	Cisco Nexus 2348 FEX
	Cisco Nexus 9000
Firewalls	Cisco ASA 55xx
Controladoras *wireless*	Cisco 8510 / 2504
UPS (*Uninterruptible Power Supply*)	APC / Riello

Neste sentido, é essencial avaliar a forma como todo o fluxo de tráfego ocorre na rede (topologia lógica), mediante os equipamentos ativos que a constituem. Paralelamente, todos estes equipamentos recorrem a diversos protocolos e tecnologias, onde serão descritas com maior detalhe no subcapítulo 3.2.2.

3.2.2 Arquitetura lógica

A representação dos fluxos de tráfego de uma rede define a sua arquitetura lógica. O comportamento de uma rede de dados assenta então em diferentes protocolos e tecnologias, os quais, permitem que os sistemas integrantes consigam alcançar um melhor desempenho, mediante as necessidades dos utilizadores.

No caso em particular, existem inúmeros conceitos importantes para definir, de forma genérica, o funcionamento lógico. A rede de núcleo e distribuição da U. Porto foi aprovisionada tendo em conta os múltiplos serviços (redes) a serem transportados, de forma a disponibilizar a conetividade entre múltiplas UOs.

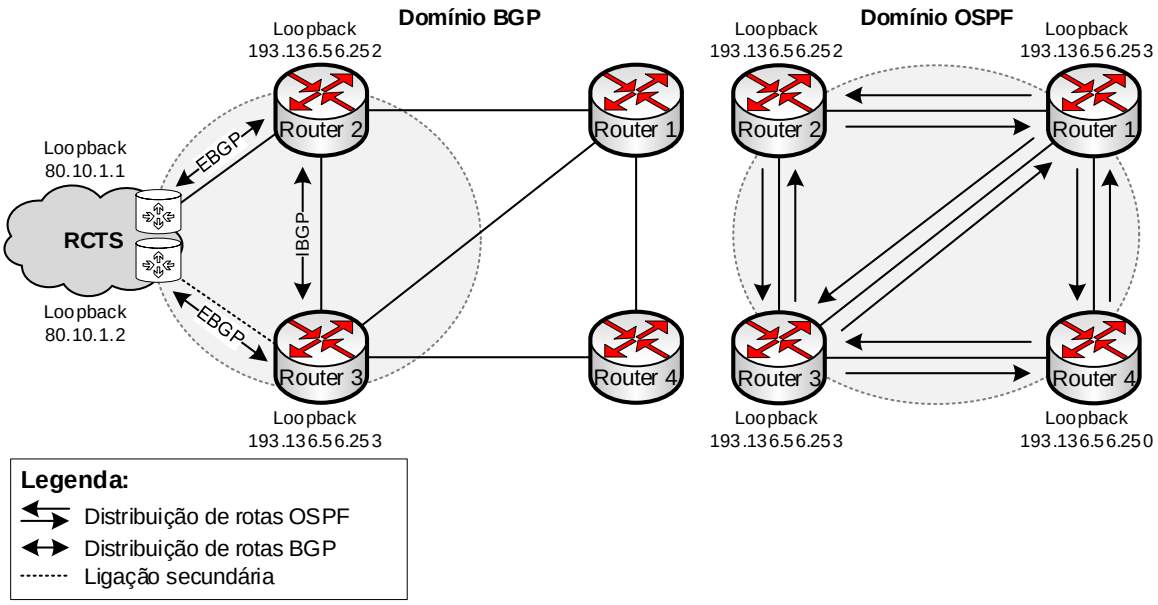


Figura 23 - Protocolos de distribuição de rotas

Em termos de rede IP tradicional, existem dois protocolos de encaminhamento em operação, cada qual com funções distintas. Dentro do domínio local, é utilizado o protocolo OSPF (*Open Shortest Path First*) [52], com o objetivo de permitir que os nós da rede contenham as rotas dinamicamente e, assim, alcancem uma determinada rede IP de destino (ver Anexo IV).

Existem quatro processos OSPF para descoberta de redes, em diferentes categorias, nomeadamente:

- Endereçamento público IPv4 e IPv6;
- Gestão *wireless*;
- Gestão IP/MPLS.

Por sua vez, para o caso do encaminhamento exterior da U. Porto o protocolo BGP (*Border Gateway Protocol*) [53] , que permite a conectividade entre sistemas autónomos (*Autonomous System*, AS) (Anexo V). A título de exemplo, demonstra-se o processo genérico de distribuição de rotas para os protocolos BGP e OSPF, conforme ilustra a Figura 23.

Para garantir a conetividade com a RCTS utiliza-se protocolo BGP, o qual faculta a partilha dos prefixos de rede IP através de políticas de *routing* (ver Figura 72). Uma vez que todas as redes IP estão sumarizadas dentro do processo BGP, as mesmas são associadas a um filtro de exportação (BGP *filter out*) para que o *remote* AS (RCTS) tenha o conhecimento das mesmas,

de modo a estabelecer duas sessões EBGP (*External Border Gateway Protocol*) (ver Figura 70).

Entre os *routers* de periferia (ASR-2 e ASR-3), existe uma sessão IBGP (*Internal Border Gateway Protocol*) para que estes nós, partilhem as redes aprendidas por EBGP e tenham conhecimento ativo da topologia da rede. Em caso de falha, a rota *default* (BGP *filter in*) será atualizada na tabela de *routing* do nó redundante (ASR-3), tendo esta como "novo" *next-hop*, o endereço remoto da rede de interligação secundária.

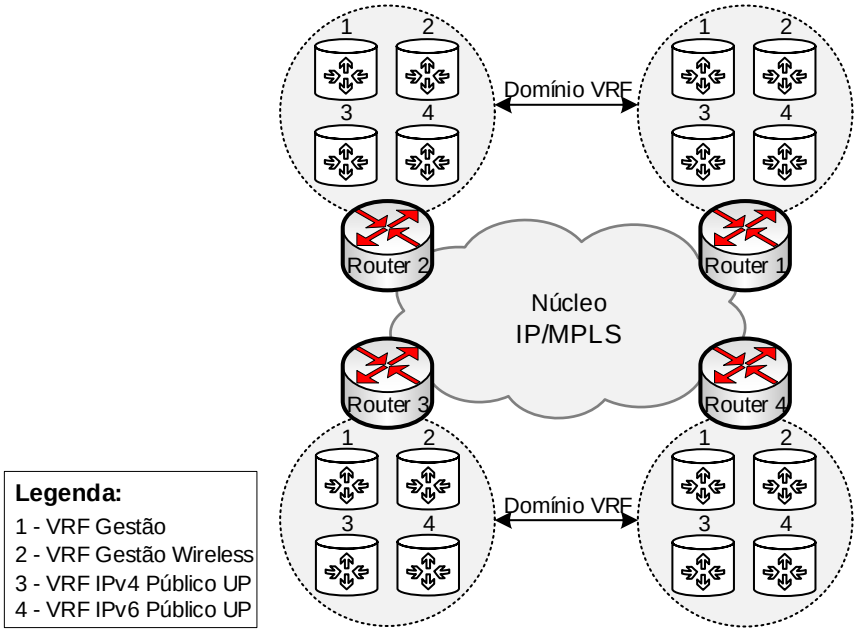


Figura 24 - Domínios VRF

Tendo em conta os protocolos de encaminhamento anteriormente referidos, salienta-se a utilização de uma tecnologia de encaminhamento virtual (*Virtual Routing and Forwarding*, VRF), que faculta a criação de comutadores IP lógicos, no mesmo comutador IP físico.

Cada VRF existente segrega diferentes domínios *broadcast*, utilizando assim diferentes interfaces lógicas de *layer* 3 (*Bridge Virtual Interface*, BVI) que possibilitam o acesso por qualquer interface física do equipamento com recurso ao MPLS (Figura 24).

Através do conceito de comutação multicamada baseada em etiquetas (MPLS), a rede da U. Porto, alia as técnicas de encaminhamento da camada de rede, a técnicas provenientes da camada de ligação de dados [54].

De um ponto de vista teórico o MPLS tem diferentes características para o seu normal funcionamento. Como principal vantagem o MPLS diminui as limitações de latência entre nós, dada a complexidade de operações das tabelas de encaminhamento (*routing table*).

No modelo de funcionamento do MPLS preconiza-se a necessidade de um protocolo de IGP (*Interior Gateway Protocol*) e um protocolo de sinalização (Anexo VI). Concretamente, nesta componente, o protocolo OSPF foi o escolhido para garantir a distribuição dinâmica de redes e o protocolo LDP (*Label Distribution Protocol*) sinaliza a distribuição de etiquetas (ver Figura 75).

O processo do OSPF configurado para descoberta dos *routers* ASR (ver Figura 74), diferencia-se e ganha um enorme relevo para todas as VRFs existentes pois, sem este, nenhuma VRF funcionaria. Isto revela uma enorme dependência hierárquica da rede, visto que toda a lógica de configuração depende da utilização do MPLS e consequentemente, dos serviços *l2vpn* disponibilizados pelo VPLS.

Qualquer recálculo na instância OSPF da rede de gestão dos *routers* ASR, implica uma alteração de "caminhos" em todas as VRFs presentes. Por outro lado, as VRFs, não necessitam de conter endereços de interligação entre os *peers* para atingir um dado prefixo de rede, pois cada nó utiliza a interface BVI como *router-id*, estando estas no mesmo domínio *broadcast*.

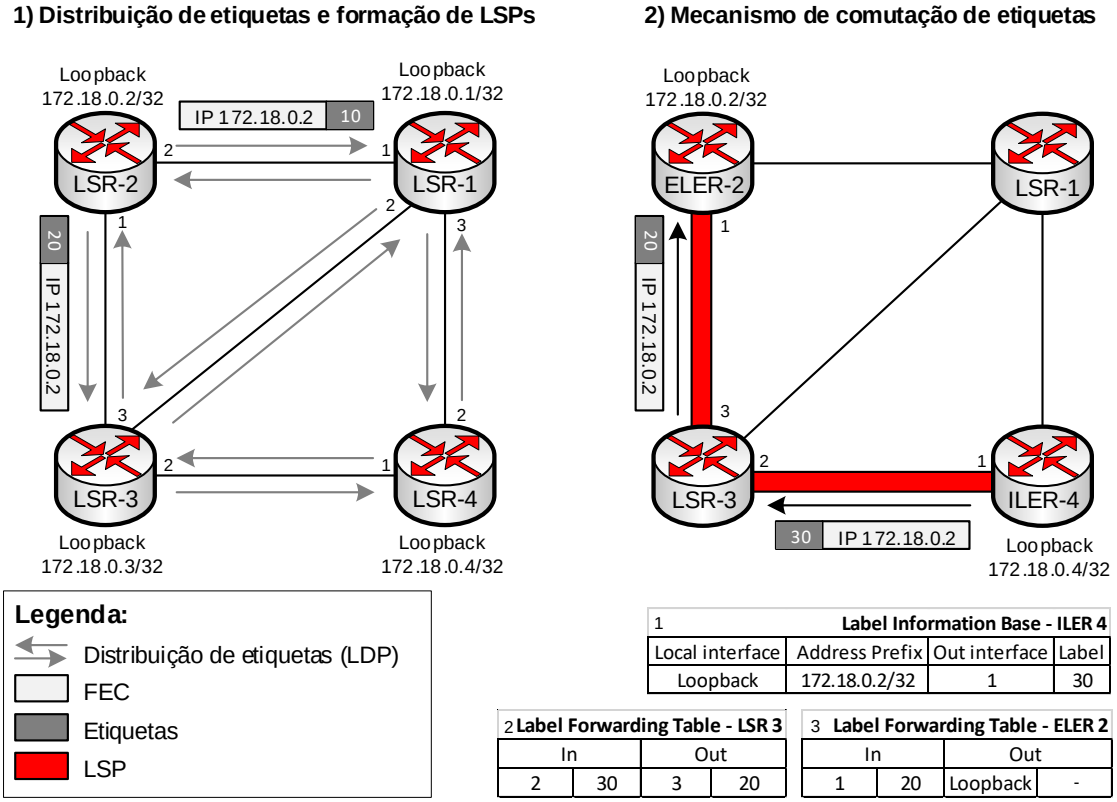


Figura 25 - Domínio MPLS

Isto deve-se ao facto das interfaces das redes de interligação estarem associados ao processo OSPF nativo ao MPLS e, intrinsecamente, as interfaces BVI funcionarem como "ponte" entre a camada 2 e a camada 3 do modelo TCP/IP. Em paralelo a isso, as interfaces BVI são agregadas a serviços *l2vpn* (*routed interface*) de modo a fornecerem a ligação intracamadas.

A arquitetura MPLS define diferentes nomenclaturas aos nós de rede. Os nós LER (*Label Edge Router*) localizam-se na periferia da rede e compreendem as tarefas mais complexas, como a interligação de redes, a classificação de pacotes e a determinação da sua FEC (*Forwarding equivalence class*). No interior da rede os nós LSR (*Label Switching Router*) encaminham os pacotes, baseando-se apenas no valor da etiqueta e na respetiva tabela de comutação (*Label Forwarding Information Base*, LFIB).

Na Figura 25 está representado o conceito fundamental do MPLS. Numa fase inicial, são distribuídas as etiquetas entre todos os nós (*hop by hop*), sendo este processo iniciado pelo nó a jusante, mediante a origem e destino do fluxo de dados (LSR-2, LSR-4).

Tendo em conta que todos os nós da topologia MPLS já redistribuíram todas as FECs associadas às respetivas etiquetas, os caminhos (*Label Switched Path*, LSP), são então estabelecidos unidirecionalmente entre todas as interfaces pertencentes ao domínio MPLS. A partir deste ponto, o *Ingress* LER (ILER4) determina a etiqueta a adicionar ao pacote e encaminha-o para a interface correspondente de saída. O LSR-3, com auxílio do LSP anteriormente estabelecido, pode agora receber o pacote e comutar apenas com base na etiqueta. Por fim o *Egress* LER (ELER-2) tem a responsabilidade de remover a etiqueta MPLS antes de entregar ao seu destinatário.

Embora o MPLS retrate a possibilidade de distribuir redes associadas a diferentes FECs para a respetiva comutação de etiquetas, o modelo MPLS subsistente, não utiliza esse conceito na distribuição das redes provenientes das UOs (LAN). Apenas as redes de interligação entre os *routers* e os endereços IP de *loopback* fazem uso do conceito supracitado, de modo a estabelecerem os LSPs.

Inerente a esta tecnologia, os serviços VPLS empregues, possibilitam que a rede se comporte como um comutador *Ethernet* virtual (*layer* 2 do modelo TCP/IP) [55]. Todos os LSPs previamente estabelecidos pelo modelo MPLS implementado são então utilizados pela tecnologia VPLS, viabilizando assim o transporte de múltiplos serviços. A descoberta automatizada dos *neighbors* para instanciação de novos serviços *l2vpn*, tem por base o mecanismo BGP *Auto Discovery* (ver Figura 76 e Figura 77).

Os elementos de rede MPLS, anteriormente citados, sobre o ponto de vista do modelo de referência VPLS, adquirem assim novas nomenclaturas (Figura 26).

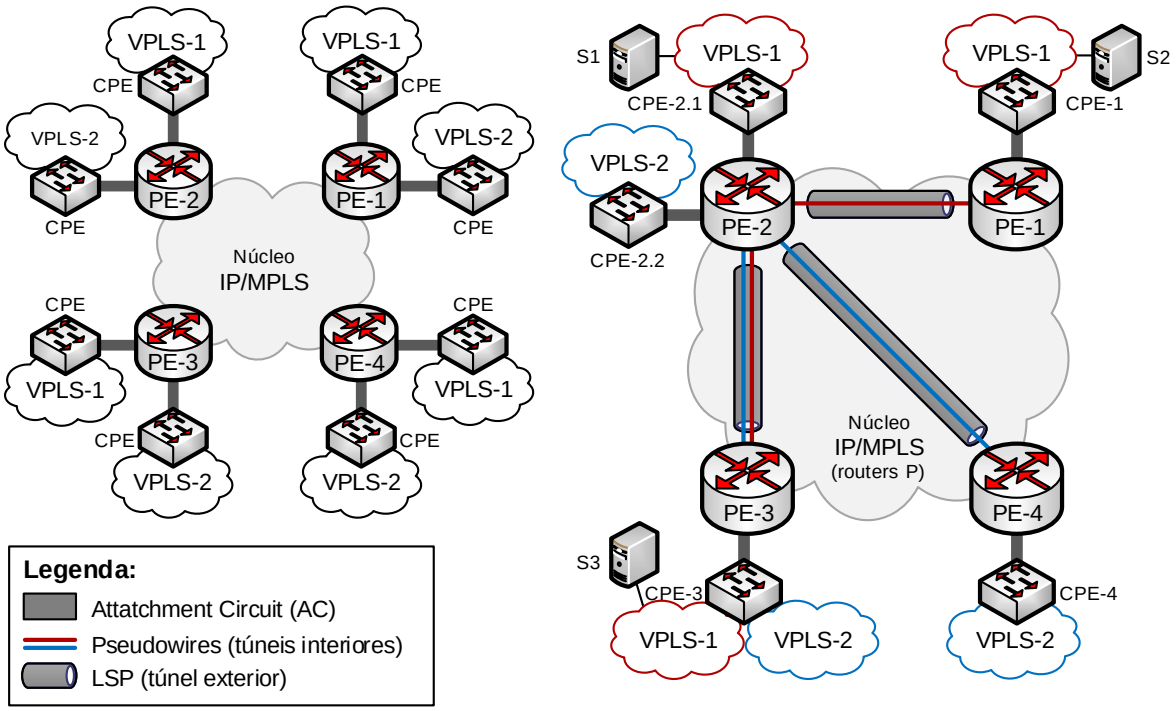


Figura 26 - Exemplo do modelo de referência VPLS

Os comutadores *Ethernet* localizados nos *datacenters* ou nas instalações das diferentes UOs são denominados por *Costumer Provider Edge* (CPE), estando estes interligados a PEs (*Provider Edges*) através de um circuito de acesso (*Attachment Circuits*, AC). Em relação aos nós de acesso ao domínio MPLS, estes são intitulados de PEs e têm como função principal determinar a origem e o término de uma ou mais instâncias VPLS (serviços). Associados ao interior da nuvem IP/MPLS, existem na sua constituição *routers* P (*Provider*), os quais são incumbidos por interligar PEs e transportar os túneis interiores (*pseudowires*) e exteriores (LSPs) necessários ao funcionamento do VPLS.

Com o auxílio da Figura 26, analisemos então um exemplo de um serviço VPLS transportado na rede de núcleo da U. Porto. A título meramente ilustrativo, iremos abordar um serviço multiponto que permite a conectividade unidirecional entre três servidores que disponibilizam serviços de monitorização alarmística com base na ferramenta Nagios.

Supondo que duas tramas *Ethernet* são emitidas por um CPE de periferia (CPE-2.1) até aos respetivos CPEs de destino (CPE-3, CPE-1), quando o PE-2 recebe as tramas transmitidas originalmente pelo servidor (S1), determina o serviço VPLS (VPLS-1) em causa através do circuito de ligação (AC) e da etiqueta 802.1Q da trama (VLAN) onde esta foi recebida.

Tendo como princípio orientador de que os PEs já têm os LSPs estabelecidos com base nas etiquetas exteriores (*tunnel-label*) provenientes do MPLS, são então adicionadas duas etiquetas (processo de *push*) pelo PE-2 às tramas iniciais, as quais identificam o *pseudowire* respetivo (PW-*label*).

Quando os pacotes circulam no núcleo MPLS/IP, são encaminhados diretamente para os PEs de destino (PE-1, PE-3), visto que neste exemplo não existe o conceito teórico de *routers* P. Ainda assim, ao analisarmos a Figura 26, podemos verificar um caso em que esse conceito existe. Um determinado pacote referente à instância VPLS-2 terá de ser encaminhado no núcleo MPLS/IP pelos mecanismos de *label switching/swapping* para alcançar um dado destino no PE-4.

Ao alcançar os PEs de destino (PE-1, PE-3), o desencapsulamento dos pacotes é então realizado. Após as etiquetas exteriores serem retiradas (processo de *popping*), a instância VPLS-1 é identificada com base na etiqueta interior. Por fim, as tramas são enviadas pelos PEs (PE-1, PE-3) aos CPEs (CE-3, CE-4) que interligam os servidores de destino (S2, S3).

3.3 Rede de gestão

A dimensão e a escalabilidade de uma rede depende no geral, da sua infraestrutura de gestão e esta prossupõe critérios de ordem económica, tecnológica, funcional e política [3]. Tais critérios inviabilizam, muitas das vezes, o desenvolvimento de uma rede de gestão totalmente dedicada que cumpra com um cenário ideal, nomeadamente no que respeita às normas de resiliência e transparência do tráfego de gestão (gestão *Out-of-Band* e gestão *in-Band*).

Os serviços de conectividade são então um dos principais requisitos de uma rede de comunicação de dados e, em consequência disso, a estruturação de uma rede de gestão deve respeitar o menor impacto possível nos mesmos.

Posto isto, de modo a supervisionar e controlar todos os equipamentos de rede que se encontram nesse domínio, dividiu-se a rede de gestão em três camadas constituintes, nomeadamente:

- Mecanismos de segurança;
- Rede de gestão de equipamentos;
- Rede de gestão *wireless*

3.3.1 Gestão *out-of-band* e *in-band*

O conceito de gestão fora de banda (OOBM) retrata metodologias de estruturação de rede, ao nível físico e lógico, que proporcionam a dissociação do tráfego de gestão (*management plane*) e do tráfego de dados (*data plane*). Em contrapartida a gestão realizada dentro de banda (IBM), não garante esta diferenciação.

Na topologia de rede da U. Porto, não está implementada a metodologia de gestão OOB. Atualmente os equipamentos de rede presentes dispõem de interfaces de gestão dedicadas (porta de consola ou porta *ethernet*), as quais possibilitam criar uma VLAN de gestão para configurar e administrar os equipamentos. As interfaces de gestão dedicadas, por si só, não cumprem com o conceito OOBM na sua totalidade.

Numa primeira abordagem, podemos constatar a falta de equipamentos dedicados apenas à gestão OOB, não havendo assim uma rede física totalmente separada da rede de dados comum. Por outro lado, os equipamentos, na sua grande maioria, não têm uma interface de gestão dedicada em utilização. Os equipamentos de rede, de forma global, são então geridos pelas interfaces comuns à rede de serviços, ou seja, pelo método de gestão IB.

Embora existam ligações físicas comuns entre os equipamentos, as VLANs acabam por oferecer um isolamento do tráfego de gestão. Ao nível da gestão IB é importante salientar que a redundância de ligações é efetuada por recurso ao conceito de *port-channels*, associando assim, duas ou mais interfaces físicas do equipamento com a respetiva VLAN de gestão.

3.3.2 Mecanismos de segurança

A rede de gestão de equipamentos da U. Porto está dotada de mecanismos de segurança que, de forma transversal, garantem o acesso a todos os elementos que a compõem, cumprindo com os principais requisitos de segurança de redes de comunicação, nomeadamente no que respeita à autenticação, confidencialidade, integridade, controlo de acessos e disponibilidade.

Devido à necessidade de restringir o acesso à rede de gestão, existiu o dever de controlar os acessos dos utilizadores à rede. Assim sendo, apenas os administradores da rede devem aceder à rede de gestão de equipamentos, para assim gerir e administrar toda a infraestrutura existente.

Para controlar os acessos à rede, são utilizadas *firewalls*, com o objetivo de proteger a rede privada da organização, de acessos não autorizados oriundos da *internet*, ou para proteger uma dada rede de acessos a partir de qualquer outra rede.

Os tipos de *firewalls* utilizadas na U. Porto, baseiam-se em filtros de pacotes (*packet filters*) e na inspeção do estado destes (*stateful inspection*). Estas *firewalls* analisam o tráfego aos níveis IP e TCP/UDP (camada 3 e 4 do modelo TCP/IP). A decisão de encaminhamento de um determinado pacote é realizada com base na configuração de listas de controlo de acessos (*Access Control Lists*, ACL).

As metodologias de acesso externo à rede de gestão são bastante restritas, sendo utilizados canais de comunicação seguros e virtualmente dedicados, em particular, as redes privadas virtuais (*Virtual Private Networks*, VPN) [56].

Destaca-se assim, o IPSec (IP *Security*) como a tecnologia empregue pela U. Porto (Anexo VIII) para garantir a comunicação segura entre redes privadas interligadas por circuitos/canais virtuais suportados noutras redes (públicas). O IPSec funciona ao nível de rede, sendo completamente invisível do ponto de vista do utilizador. Examinemos então os dois casos de configuração da arquitetura IPSec presente na infraestrutura da rede de gestão (Figura 27):

- No modelo de comunicação IPSec entre o utilizador e a *firewall* (*remote-to-access*) de gestão (interface *outside*), o acesso é realizado com recurso a uma ligação estabelecida através da *internet* facultando o acesso remoto, de modo seguro, às redes privadas (interfaces *inside*) de gestão (Anexo VIII - 2.);
- O modelo de comunicação entre redes (*site-to-site*), providencia um túnel lógico entre duas *firewalls* (interfaces *outside*), melhorando a segurança do tráfego de gestão proveniente das redes internas e transportado sobre as redes públicas da U. Porto (Anexo VIII - 1.).

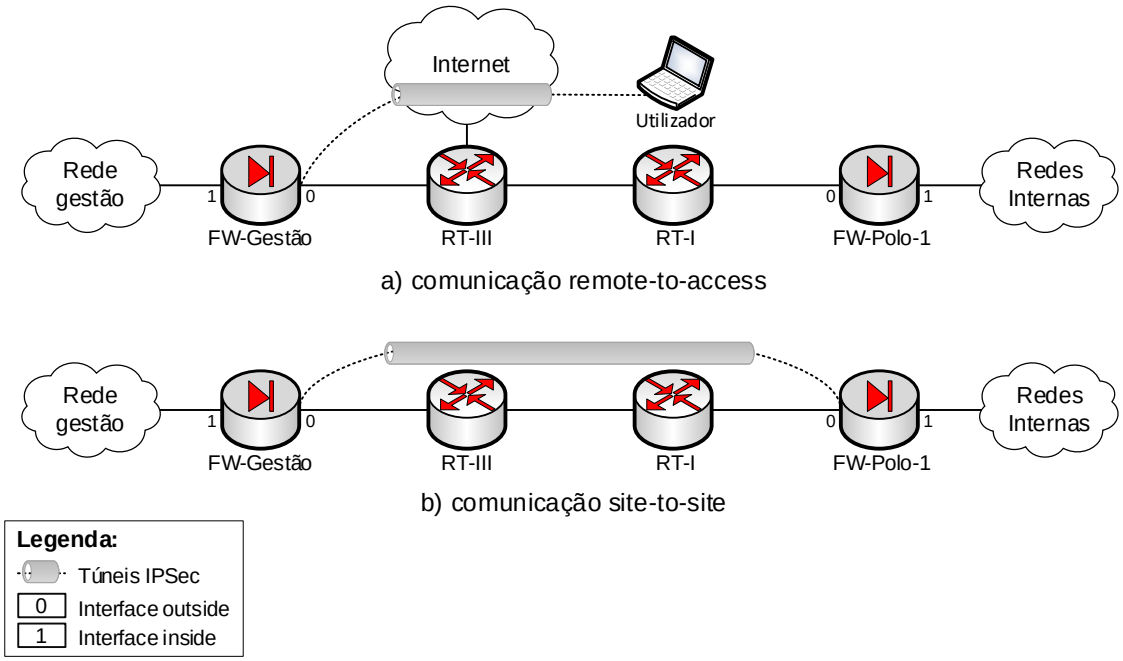


Figura 27 - Comunicação em ambiente IPSec

Em termos de segurança, diversos aspetos são acautelados pelas VPNs em particular. A confidencialidade é garantida por encriptação e *tunnelling* utilizando-se, no primeiro caso, mecanismos de encriptação de pacotes (3DES, *Triple Data Encryption Standard* e AES, *Advanced Encryption Standard*) e no segundo, normas como o IPSec e o L2TP (*Layer 2 Tunneling Protocol*).

Para além da confidencialidade, existem configurados mecanismos de integridade através de algoritmos de *hashing*, como o SHA (*Secure Hash Algorithms*) e o MD5 (*Message-Digest algorithm 5*). Como solução genérica para assegurar a segurança sobre as ligações TCP, com base nos mecanismos anteriormente referidos, é empregue como suporte protocolar o SSL/TLS (*Secure Socket Layer/Transport Layer Security*).

No que respeita à autenticação, esta visa estabelecer a entidade de um utilizador e/ou sistema, tendo em vista a determinação de ações permitidas (autorização), com base no controlo de acessos aos recursos e posterior contabilização.

Deste modo, as VPNs configuradas, não analisam por si só os padrões inerentes ao polinómio AAA (*Authentication, Authorization, Accounting*).

Para determinar que utilizadores devem ter acesso à camada de gestão, são empregues servidores que utilizam protocolos de AAA. Podemos definir que existem três principais grupos de utilizadores na U. Porto:

- Utilizadores que não fazem parte da U. Porto, mas carecem de acesso à rede, por motivos de projetos de colaboração externa, entre outros;
- Utilizadores que não pertencem à U. Porto, porém necessitam de se autenticar na rede sem fios *eduroam* (rede de serviços internacionais de *roaming*);
- Utilizadores afetos à U. Porto.

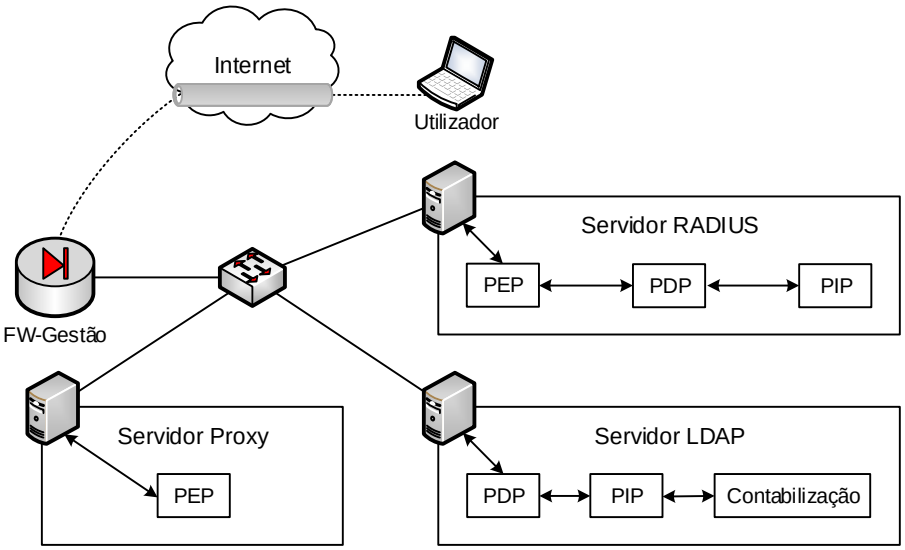


Figura 28 - Componentes do sistema de AAA da U. Porto

Mediante um utilizador que se autentique na rede, diferentes componentes dos sistemas AAA são acionados. Se um dado utilizador não pertencer à U. Porto e for estritamente necessário ter acesso à rede, o mesmo é autenticado (*Policy Enforcement Point*, PEP) pelo servidor RADIUS. A determinação de validação da autenticação e autorização é tomada pela componente de decisão (*Policy Decision Point*, PDP). Como o utilizador não tem credenciais de autenticação no repositório (*Policy Information Point*, PIP) da U. Porto, as mesmas têm de ser criadas no servidor RADIUS e guardadas localmente [57] (Figura 28).

Por outro lado, se um utilizador for administrador de rede da U. Porto, as suas credenciais existem num repositório comum nomenclado por PEP, estando este integrado no servidor RADIUS para proceder à autenticação, sendo o processo de autorização e gestão de credenciais validado pelo servidor LDAP (*Lightweight Directory Access Protocol*) [58]. Determinados privilégios do utilizador, podem ser contabilizados por um repositório específico para o efeito. Caso um utilizador não necessite de acesso à rede e apenas careça de serviços de conectividade através da rede sem fios *eduroam*, o seu processo de autenticação é realizado com recurso a um servidor *proxy*, o qual reencaminha os pedidos inerentes ao modelo AAA, para a instituição a que este pertence (Figura 28).

Os critérios de acesso interno à rede de gestão por parte dos grupos de utilizadores previamente citados, funcionam de igual forma, com exceção aos utilizadores que, de algum modo, fazem a manutenção dos serviços de conectividade da rede, podendo estes ter credenciais de acesso direto aos equipamentos, não utilizando assim os mecanismos de AAA para a respetiva autenticação. Utilizadores que administram a rede de comunicações e os sistemas da U. Porto, estão então diretamente ligados à rede de gestão, com acesso às redes e portas configuradas nas ACLs das *firewalls*.

3.3.3 Gestão de equipamentos

A rede de gestão de equipamentos pode ser vista de forma genérica, como uma rede que permite gerir a configuração dos mesmos, cumprindo com os requisitos de segurança no que visa ao controlo e acesso dos dispositivos.

Face à necessidade de gerir alguns equipamentos que pertencem às UOs, são utilizadas redes de interligação entre estas e as *firewalls* de gestão da U. Porto. Isto permite não só utilizar endereçamento IP privado, como poupar endereçamento público, diminuindo assim, a carência de recorrer a mecanismos de tradução de endereços de rede (*Network Address Translation*, NAT).

Na infraestrutura de rede de núcleo, estão configuradas nos *routers*, interfaces lógicas de *layer* 2, de modo a dissociar o tráfego de gestão, fazendo o uso das tecnologias (VPLS sobre IP/MPLS) previamente descritas (consultar secção 3.2.2). Cada *datacenter* por pólo, contém uma rede de gestão de equipamentos dedicada que pertence a uma VRF de gestão associada aos *routers* (RT).

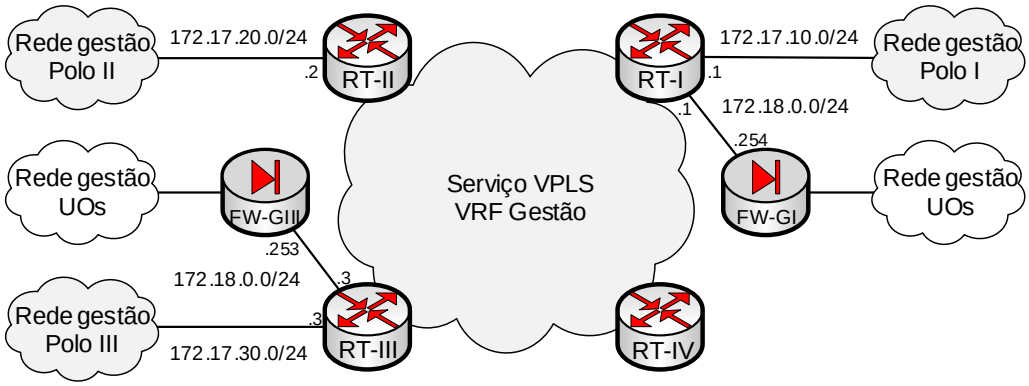


Figura 29 - Redes de gestão de equipamentos

Com a utilização da tecnologia *Ethernet* (VPLS), torna-se possível instanciar diferentes serviços por cada rede de gestão que se deseja transportar nos equipamentos de núcleo, garantindo também a redundância de caminhos, de forma mais simples e eficaz. Cada *router* possui uma interface virtual de *layer* 3, servindo esta de *default gateway* para a rede de gestão de pólo, à qual este pertence. Sempre que uma rede na VRF de gestão não seja conhecida por um determinado *router*, este encaminha o tráfego por uma rota *default* com o endereço de destino da subinterface da *firewall* de gestão (Figura 29).

Em termos de redundância das *firewalls* de gestão, existe um serviço VPLS configurado especificamente para o mecanismo de tolerância a falhas (*failover*) [59], viabilizando a comunicação e o processo de troca de configurações. A *default gateway* da rede de gestão que se encontra em uso é sempre atribuída à *firewall* primária, assim sendo a *firewall* secundária (inativa) contém um endereço diferente na sua subinterface, o qual nunca é conhecido pelos equipamentos de núcleo. Em caso de falha, uma das *firewalls* assume então o endereço primário configurado na sua subinterface de gestão, sendo este processo transparente aos restantes equipamentos.

3.3.4 Gestão de rede *wireless*

À semelhança da infraestrutura da rede de gestão de equipamentos, a rede de gestão sem fios (*wireless*), contém uma VRF de gestão *wireless*, cumprindo de igual modo com as mesmas configurações ao nível *routing* da rede de núcleo (*routers*), conforme anteriormente referido (secção 3.3.3).

A grande diferença da rede de gestão *wireless*, prende-se então ao facto de como esta se encontra, de forma a facilitar o controlo e a administração de todos os pontos de acesso (*Acess-Point*, AP) sem fios.

Devido à transversalidade dos serviços de conectividade da rede *eduroam*, a lógica de configuração da rede de gestão *wireless*, depende muito dos equipamentos de núcleo e distribuição e impõe procedimentos que garantam o acesso às UOs através de serviços VPLS.

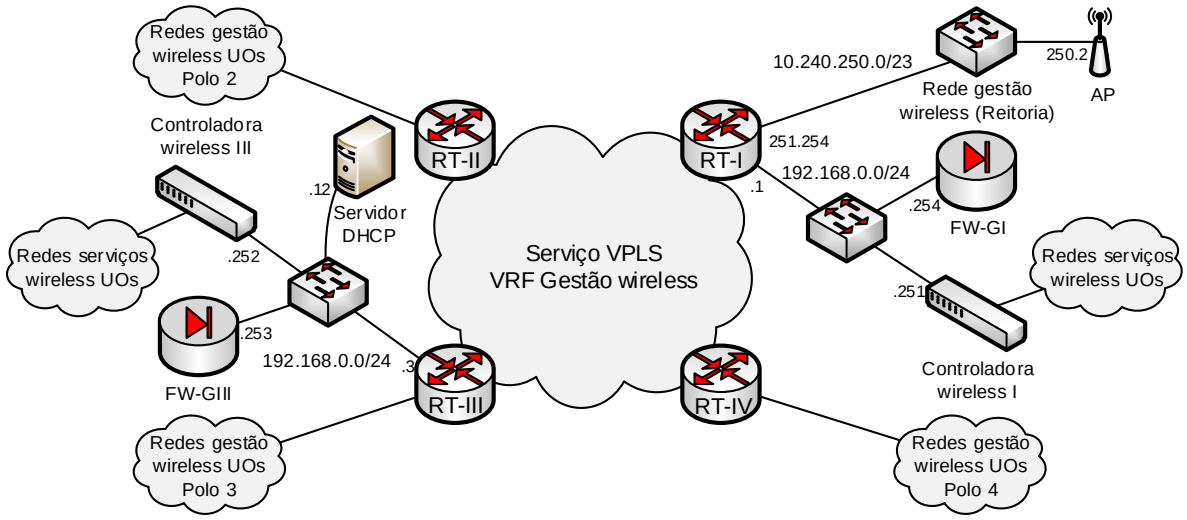


Figura 30 - Rede de gestão *wireless*

Tendo como referência a Figura 30, fica percetível os principais equipamentos que decompõem a rede de gestão *wireless* da U. Porto. Todos os APs pertencentes às UOs necessitam de endereçamento IP, sendo este atribuído dinamicamente por um servidor DHCP [60]. O processo de autenticação para atribuição de endereços IP, por parte do servidor DHCP, é então efetuado com a validação do endereço MAC de um AP em particular. As controladoras *wireless* são responsáveis pela descoberta dos APs, bem como a respetiva configuração dos mesmos, nomeadamente no concerne à difusão dos diferentes SSIDs (*Service Set Identifier*) das redes de serviços *wireless*.

Analisemos então um caso em concreto do modelo de gestão *wireless* preconizado na U. Porto. Considerando que um determinado ponto de acesso (AP) sem fios foi instalado no edifício da Reitoria, torna-se importante salientar como este fica alcançável através da rede de gestão, de modo a ser configurável através da controladora *wireless*.

Considerando que o endereço MAC do AP foi adicionado ao ficheiro do servidor DHCP, respetivo ao endereçamento de rede da Reitoria, o ponto de acesso sem fios (cliente DHCP) emite um pedido DHCP de descoberta (*discover*) com destino ao servidor DHCP, encaminhando-o para a sua *defaut gateway* (10.240.251.254). Os prefixos de rede respeitantes às UOs estão associados à VRF de gestão *wireless*, onde está definida uma rota *default* com destino ao endereço IP da *firewall* de gestão (192.168.0.254). Neste caso, o endereço de destino

do servidor DHCP é conhecido pelo *router* RT-I, estando este configurado na própria interface de rede de onde o pedido DHCP é originado (*helper address*).

Como o servidor DHCP (192.168.0.12) não está na mesma rede de gestão *wireless* da Reitoria, são utilizados mecanismos de DHCP *relay* na interface de onde o pedido DHCP é proveniente, permitindo assim todos os pedidos DHCP transitarem entre as duas redes. O *router* RT-I (*agente relay*) encaminha então o pedido DHCP para a interface da rede (192.168.0.1). Esta interface está associada ao serviço VPLS existente viabilizando que o domínio *broadcast* seja estendido até ao *router* do Pólo 3 (RT-III).

O *router* RT-III como contém uma interface de *layer* 3 (192.168.0.3) na mesma rede de gestão *wireless* do servidor DHCP, encaminha o pedido DHCP de descoberta até ao servidor DHCP. Em resposta a este pedido, o servidor DHCP envia uma mensagem de oferta (*offer*), com as configurações de rede. Segundo os mecanismos VPLS, anteriormente abordados (secção 3.2.2), a mensagem de oferta é então recebida pelo AP. Por sua vez, o AP responde ao servidor DHCP com uma mensagem (*request*), a solicitar o IP anteriormente disponibilizado. Por fim o servidor DHCP envia uma mensagem de confirmação (*acknowledge*) ao AP, com a reserva do endereço IP a ser utilizado, bem como o tempo pelo qual este é válido.

A partir deste momento, o AP tem um endereço IP pertencente à rede de gestão da Reitoria. No ficheiro do servidor DHCP são realizados os mapeamentos entre os IPs e os endereços MAC dos APs, estando também configurados, os endereços MAC das controladoras *wireless* (I ou III, mediante opção).

Para que o AP (modo *Lightweight*) consiga descobrir as controladoras WLC (*Wireless* LAN *Controller*) e assim carregar a configuração pela primeira vez, envia pedidos (em *unicast*) através do algoritmo de descoberta (*Lightweight Access Point Protocol*, LWAPP), criando uma lista interna das controladoras detetadas na rede [61]. Esta descoberta é realizada na camada de acesso à rede com recurso aos endereços MAC das interfaces de gestão das controladoras. Por norma a controladora *wireless* (I) é sempre aquela que está especificada como primária nos ficheiros do servidor DHCP (*option* 43) [62].

Após este processo a controladora verifica se o AP tem a imagem com a configuração da controladora. No caso concreto, este passo não se verifica. O AP descarrega então a imagem da controladora *wireless* (192.168.0.251) e reinicia de forma a instalar a configuração *default*.

Posteriormente acedendo à controladora, é possível definir uma lista de controladoras *wireless* a que o AP pertence, sendo esta descrita por ordem de preferência. Todas as

parametrizações dos APs são realizadas nas controladoras *wireless*. As interfaces de serviço são configuradas diretamente nas controladoras, garantindo assim que os APs difundam os serviços de conectividade.

A título de curiosidade salienta-se também que os equipamentos de rede de cada UO são *gateway* das redes de serviços existentes nas controladoras *wireless*, possibilitando que o tráfego de saída não seja encaminhado pela controladora.

3.4 Gestão operacional

O plano de gestão operacional visa conceptualizar métodos operacionais, sejam eles recursos humanos, materiais ou tecnológicos, que permitem gerir e administrar uma organização de forma eficaz, sem comprometer a sua produtividade.

A indisponibilidade dos serviços de IT (*Information Technology*) pode originar múltiplos problemas, que por vezes se traduzem na perda de informação relevante, prejudicando assim a missão da instituição.

O planeamento da gestão operacional da rede da U. Porto engloba metodologias de modo a assegurar os serviços críticos e a otimização das tarefas desenvolvidas diariamente. Tais procedimentos devem ser retratados, compreendendo conceitos fundamentais que apresentam não só a automatização destes processos, como os meios tecnológicos que os sustentam.

A fim de organizar um conjunto de ferramentas, procedimentos, políticas e atividades realizadas pela U. Porto, torna-se importante evidenciar o modelo funcional FCAPS (secção 2.3.4).

Concretamente os sistemas de monitorização dos equipamentos e serviços de IT, os quais dão suporte à infraestrutura de rede da U. Porto, irão ser descritos de acordo com a estrutura recomendada pelo FCAPS, facilitando assim o melhor entendimento da gestão operacional em vigor.

3.4.1 Infraestrutura de gestão FCAPS

Toda a infraestrutura física que apoia a gestão e monitorização da rede de dados da U. Porto compreende os diferentes paradigmas e contextos computacionais preconizados pelo FCAPS (Tabela 9). Não só pelo princípio da resiliência, mas também pelo cumprimento de uma gestão autónoma dos sistemas virtualizados, os servidores centrais acumulam as principais áreas do modelo FCAPS.

Tabela 9 - Sistemas informáticos de gestão e monitorização da U. Porto

Sistema computacional	Monitorização	Aplicações	Localização física
Sistemas informáticos físicos			
FCAPS central	Gestão de falhas	Nagios	Pólo 2 e Pólo 3
	Gestão de configuração	Rancid	
	Gestão de desempenho	RRDtool / Weathermap	
	Gestão de segurança	Syslog-NG	
Grafana	Gestão de desempenho	Telegraf, InfluxDB e Grafana	Pólo 3
Sistema de virtualização KVM			
Cisco Prime	Gestão falhas, desempenho, contabilização	Cisco Prime	Pólo 1, 2 ou 3
Portal de gestão de autenticação	Gestão de segurança	*Software* interno	Pólo 1, 2 ou 3
Servidores LXC (*Linux Containers*)			
Graylog	Gestão de segurança	Graylog, mongoDB e Elasticsearch	Pólo 3

A metodologia de monitorização de equipamentos através do servidor FCAPS central (ativo) é descrita em dois conceitos topológicos diferenciados (Figura 31). Como tal, este sistema contempla duas interfaces de rede dedicadas, para uma monitorização de rede proficiente e totalmente segmentada:

- Os equipamentos informáticos provenientes da rede de serviços são monitorizados com base no endereçamento público (VRF IPV4);
- Os dispositivos alcançáveis pela rede de gestão são monitorizados com recurso ao endereçamento IP privado (VRF Gestão).

Para o cenário retratado na Figura 31 é importante reter alguns aspetos, nomeadamente o facto do servidor FCAPS central alcançar as redes de domínio público através do processo de encaminhamento tradicional IP configurado nos *routers* (BV) presentes na rede de núcleo (Anexo IV).

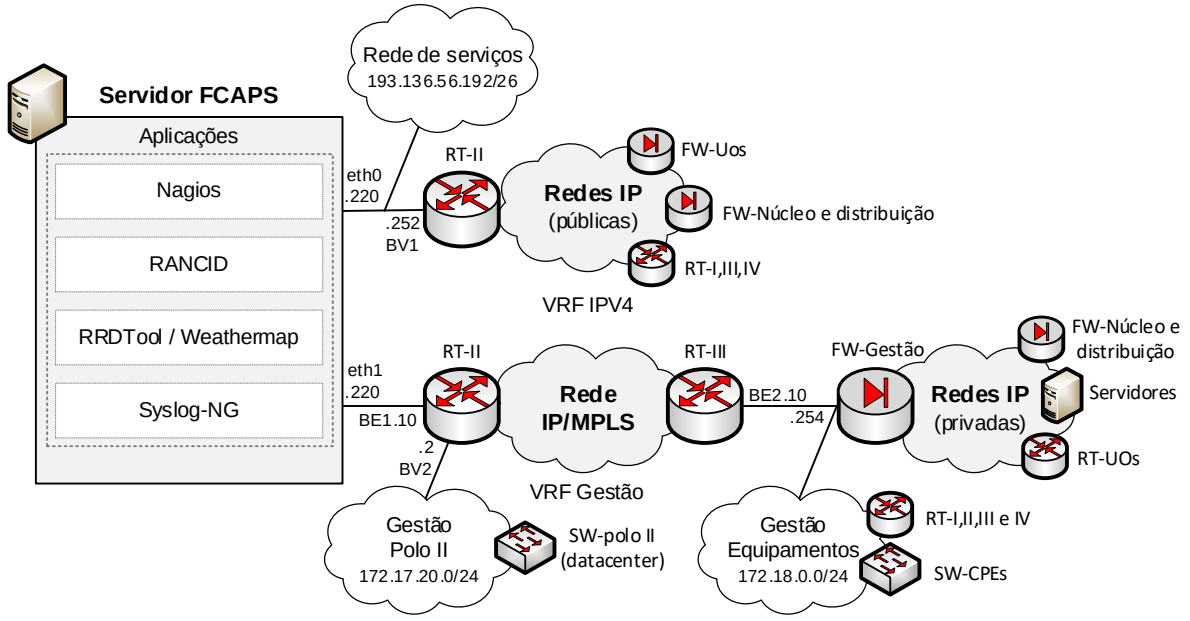


Figura 31 - Infraestrutura de rede do servidor FCAPS central

Além disso, é importante evidenciar o comportamento lógico da interface alocada à rede de gestão. Esta topologia obriga à instanciação de rotas estáticas no próprio servidor FCAPS, sendo que todo tráfego de endereçamento de IP privado é redirecionado para o domínio *broadcast* da rede de gestão de equipamentos, estando este configurado na *firewall* de gestão. Por sua vez, a rede de gestão de equipamentos (172.18.0.0/24) tem acesso às redes de gestão dos diferentes pólos, (rede de gestão Pólo 1, 2 e 3), fazendo uso dos processos de encaminhamento IP nesta *firewall*, recorrendo a tecnologias IP/MPLS para transportar os serviços VPLS no núcleo de rede e, assim, providenciar a entrega nos circuitos de acesso (*Bundle Ethernet*, BE).

Existem quatro plataformas de monitorização implementadas no servidor FCAPS central, entre as quais enumeram-se as seguintes:

- Nagios
- Rancid
- RRDTool / Weathermap
- Syslog-NG

A falha de disponibilidade do servidor FCAPS primário depende da execução manual de um *script* que tem como objetivo sincronizar as configurações diárias entre os dois servidores FCAPS atuais. A execução do *script* irá atualizar os principais ficheiros de configuração das aplicações acima mencionadas.

Ainda como sistema computacional físico dedicado, destaca-se o servidor Grafana. Este sistema informático encontra-se na rede de gestão *wireless*, com a *default-gateway* configurada na *firewall* de gestão.

Os restantes sistemas que auxiliam o apoio ao modelo FCAPS são virtualizados em duas infraestruturas próprias:

- Primeiramente destaca-se o KVM (*Kernel Virtual Machine*) como uma estrutura física que dá suporte a dois sistemas virtuais, os quais integram as plataformas Cisco Prime e Portal de gestão de autenticação;
- Refere-se em segundo lugar, o sistema baseado em Linux *Containers* (LXC) que incorpora a ferramenta aplicacional Graylog. Esta infraestrutura foi concebida e é gerida pela unidade de segurança, não fazendo parte do estudo os seus métodos de monitorização (secção 3.4.2).

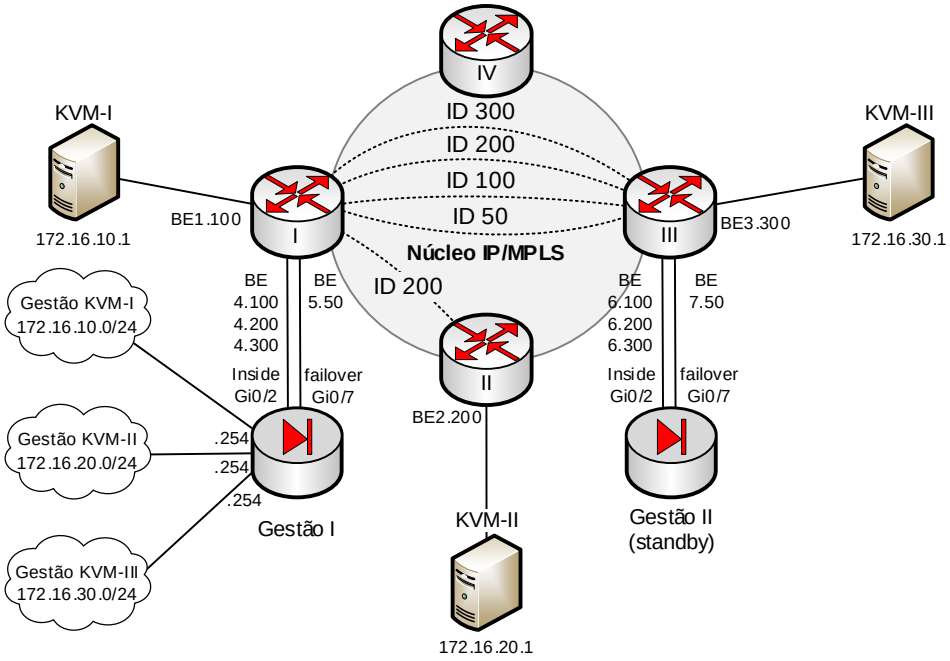


Figura 32 - Infraestrutura de rede dedicada do KVM

A título ilustrativo, podemos observar a topologia das redes de gestão do KVM alusivas a cada pólo (Figura 32). Embora geograficamente os servidores sejam distribuídos pelos respetivos pólos (*datacenters*), sob o ponto de vista lógico, as suas redes de gestão são transportadas no núcleo através dos respetivos serviços VPLS (ID) configurados e entregues nas *firewalls* de gestão (consultar Anexo VII).

Atendendo à Figura 32, podemos analisar ainda, o processo de resiliência adotado. Deste modo, todos os serviços (inclusive o serviço de *failover*) estão configurados e associados à interface dos *routers* I e III (BE4 e BE5) que providenciam a interligação com as *firewalls* de gestão (ativa e de *standby*).

Outro fator bastante importante é o facto dos sistemas virtualizados (Cisco Prime e Portal de gestão de autenticação) poderem estar localizados em qualquer sistema físico do KVM, isto é, a *default gateway* de rede de cada sistema é sempre instanciada nas *firewalls* no entanto, abrange o mesmo princípio de transporte e configuração inerentes à tecnologia VPLS.

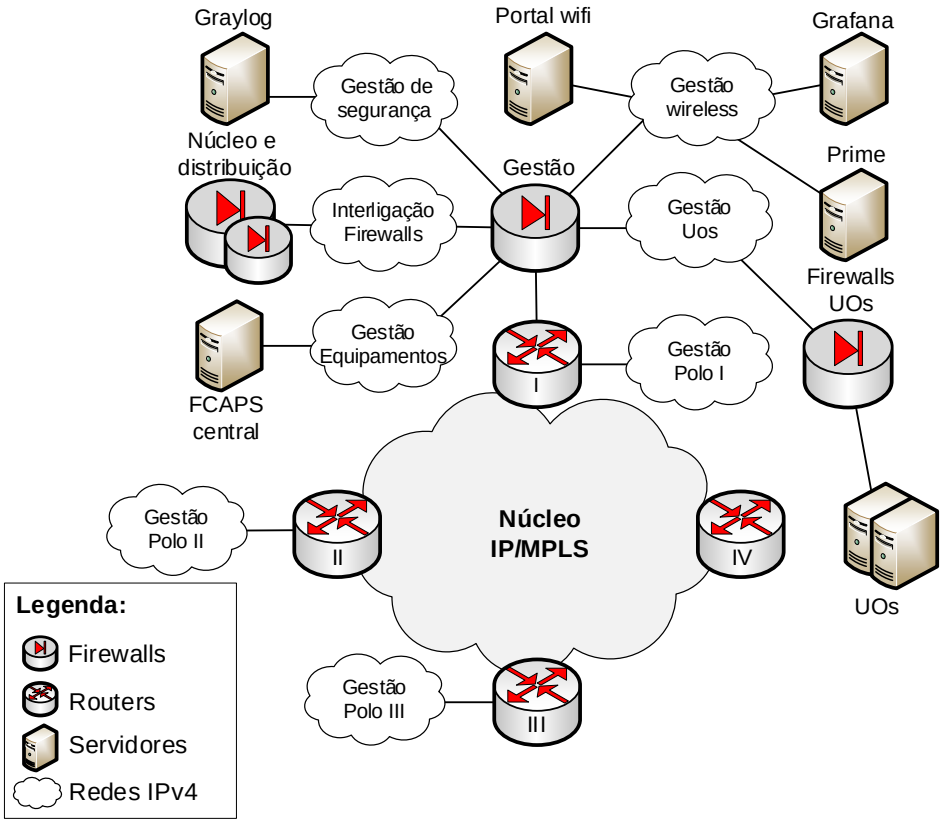


Figura 33 - Topologia lógica dos sistemas de monitorização via rede de gestão

Sintetizando, a interpretação genérica das redes de gestão intervenientes ao processo de monitorização e a localização lógica de todos os sistemas FCAPS, podem ser observadas de acordo com a Figura 33.

Para o cenário em causa devem-se reter alguns aspetos, nomeadamente a perspetiva de como todos os servidores FCAPS conseguem monitorizar a rede de dados com auxílio das redes de gestão expostas. Torna-se também fundamental salientar que as redes de gestão existentes nos *routers* de núcleo e distribuição, compreendem a gestão de equipamentos alusiva aos dispositivos de rede de acesso presentes nos *datacenters*.

No que concerne à monitorização de rede sem o recurso das redes de gestão, são realizados processos de NAT por cada sistema na *firewall* (com exceção ao servidor FCAPS central), recorrendo assim ao endereçamento público da rede de comunicação de dados IP (VRF IPV4).

3.4.2 Monitorização e gestão de redes

As ferramentas inerentes ao modelo FCAPS auxiliam as tarefas de monitorização e a gestão diária de toda a infraestrutura de rede. A base protocolar em que estas assentam descreve a informação que podemos aferir dos equipamentos de rede (Tabela 10). Os dispositivos que suportam a rede de núcleo, distribuição e acesso, não recorrem à arquitetura gestor-agente, suportando diretamente as operações de gestão que lhes são facultadas pelo sistema gestor. Tendo em conta este facto, fica percetível que apenas os sistemas computacionais fazem uso de um agente instalado e configurado no próprio terminal.

Tabela 10 - Métodos de monitorização e gestão de rede da U. Porto

Sistema gestor	Aplicações	Protocolos (operações de gestão)	Agentes
FCAPS central	Nagios	SNMP, HTTP e ICMP	NRPE
	RANCID	Telnet e SSH	
	RRDtool / Weathermap	SNMP	
	Syslog-NG	Syslog (porta 514 UDP) e TLS (porta 601 UDP)	
Grafana	Telegraf, InfluxDB e Grafana	SNMP	Telegraf
Cisco Prime	Cisco Prime	SNMP, SSHD, TFTP, HTTPS e CDP	
Portal de gestão de autenticação	*Software* proprietário	SSH, Telnet	
Graylog	Graylog, mongoDB e Elasticsearch	Syslog (porta 514 UDP) e TLS (porta 601 UDP)	

A aplicação Nagios dá suporte à monitorização alarmística, ajudando a compreender o estado geral dos sistemas constituintes de toda a rede de comunicação de dados (Tabela 11). Atualmente estão a ser monitorizados quinhentos e noventa e cinco dispositivos e mil oitocentos e quarenta e um serviços de rede. A metodologia de monitorização alarmística empregue, assenta essencialmente no protocolo SNMP e ICMP.

Tabela 11 – Equipamentos de rede monitorizados pela aplicação Nagios

Equipamentos	Modelo/Descrição	Quantidade
Routers	Cisco ASR-9010	3
	Cisco ASR-9006	1
Switches	Cisco 2950	1
	Cisco 2960	9
	Cisco 3560	2
	Cisco 4500	1
	Cisco 3600x ME	11
	Cisco Nexus 5548	8
	Cisco Nexus 9000	4
Firewalls	Cisco ASA 55xx	12
Controladoras *wireless*	Cisco 8510 / 2504	3
UPS (*Uninterruptible Power Supply*)	APC / Riello	8
Sistemas infirmáticos de gestão FCAPS	Linux	8
Dispsitivos de rede em geral	Servidores aplicacionais de serviços de rede, câmaras, réguas de energia, entre outros	524
	Total	595

A disponibilidade dos serviços nos sistemas informáticos é assegurada pelos agentes NRPE configurados localmente, criando assim um túnel criptografado no formato SSL/TLS entre o agente e o gestor (Nagios). O gestor recorre apenas ao *script* NRPE para recolher os dados dos serviços que são monitorizados pelo agente (Tabela 12).

Tabela 12 - Serviços monitorizados pela aplicação Nagios (gestor-agente)

Equipamentos	Terminologias de configuração Nagios	Verificação dos Serviços monitorizados
Sistemas informáticos	`host-alive`	Disponibilidade do sistema (ICMP)
	`check-disk`	Verifica o estado de armazenamento
	`check-load`	Valores de processos genéricos
	`check-users`	Analise e quantifica os utilizadores
	`check-cpu-load`	Valores de processamento
	`check-memory`	Analisa o estado da memória RAM

Atendendo à Tabela 13, tornam-se evidentes os principais serviços de rede que são consolidados pelo Nagios sem qualquer tipo de intermediário (agente). Os nós de rede, tal como os dispositivos que os auxiliam, retornam as métricas mediante os *scripts* definidos no gestor (`/usr/lib64/nagios/plugins`). Diante dos inúmeros *scripts* parametrizados, aos requisitos considerados mais críticos, são amplamente utilizados aqueles que garantem a disponibilidade protocolar das áreas OSPF e do domínio BGP.

Tabela 13 - Serviços monitorizados pela aplicação Nagios (gestor-cliente)

Equipamentos	Terminologias de configuração Nagios	Verificação dos Serviços monitorizados
Routers	`host-alive`	Disponibilidade
	`if-status`	Estado das interfaces
Switchs	`host-alive`	Disponibilidade
	`check-cpu`	Retorna valores de CPU
Firewalls	`host-alive`	Disponibilidade
	`check-cpu`	Estado do processador
	`failover-status`	Informação da *firewall* ativa e secundária
UPS	`host-alive`	Disponibilidade
	`if-status`	Estado das interfaces
	`temperature-status`	Retorna valores de temperatura

O Grafana foi a plataforma mais recentemente operacionalizada pela U. Porto, para colmatar a ausência de dados estatísticos, os quais eram facultados pelo RRDTool. A falta de versatilidade do RRDTool levou à introdução da nova ferramenta Grafana. A informação estatística do Grafana compreende o mesmo paradigma do Nagios, utilizando os modelos arquitetónicos, tais como, o gestor-cliente e gestor-agente.

O SNMP fundamenta a ausência do agente, retornando os dados estatísticos provenientes do tráfego que circula nas interfaces de rede dos dispositivos (Tabela 11). Em alternativa, os sistemas operacionais são monitorizados pelo agente Telegraf, o qual retorna valores para uma visualização gráfica mais detalhada.

Como elemento centralizado no auxílio da gestão de rede sem fios, a plataforma Prime, congrega os dados alarmísticos e estatísticos, monitorizando assim, as três controladoras e mil cento e oito pontos de acesso *wireless*.

Tabela 14 – Descrição dos equipamentos *wireless* da U. Porto

Equipamentos *wireless*	Fabricante / Modelo	Quantidade
Controladoras (WLC)	Cisco 8510	2
	Cisco 2504	1
Pontos de acesso sem fios (APs)	Cisco 1040	5
	Cisco 1130	4
	Cisco 1140	62
	Cisco 1240	1
	Cisco 1250	4
	Cisco 1600I	550
	Cisco 1700I	14
	Cisco 1830I	215
	Cisco 2600E	2
	Cisco 2600I	73
	Cisco 2700I	89
	Cisco 2800I	24
	Cisco 3700I	50
	Cisco 3800I	15
	Total	1111

Os dados dos principais serviços críticos são retornados ao Prime pelas controladoras *wireless*. Caso exista indisponibilidade por parte dos pontos de acesso sem fios (com base no protocolo ICMP e/ou CAPWAP), são enviados alertas para dos administradores de rede. A proficiência desta aplicação advém da extração dos diferentes valores estatísticos, estando estes representados e sustentados na área funcional em maior défice na rede cablada da U. Porto. Nesse sentido a componente de contabilização do modelo FCAPS enaltece a gestão operacional que esta ferramenta dispõe, promovendo a granularidade de dados reproduzidos ao nível do utilizador.

Os *backups* de configurações alusivos aos equipamentos de rede, são realizados pela plataforma RANCID, a qual conta com cerca de sessenta e cinco dispositivos (*routers*, *switches* e *firewalls*), estando estes enumerados ao ficheiro `router.db`. O executável `cloginrc` define os processos de autenticação para a ligação aos sistemas, de modo a extrair a configuração através dos protocolos SSH e Telnet.

Para gerir o controlo de acessos de forma centralizada, é utilizada a ferramenta Portal de gestão de autenticação. Esta plataforma foi desenvolvida e é utilizada pelos administradores de rede respeitantes à Unidade de Serviços Centrais da U. Porto. Presentemente esta aplicação dá suporte à autenticação de VPNs. Paralelamente à interligação às *firewalls*, recebe registos dos

protocolos AAA subjacentes aos servidores RADIUS, fornecendo assim o controlo dos utilizadores relativos à rede *wireless*.

O controlo de registos é suportado pela aplicação Syslog-NG, sustentando a supervisão de setenta e oito dispositivos de rede. No entanto, o sistema Graylog é atualmente a plataforma que irá examinar todos os *logs* (valores bastante elevados), conseguindo fornecer uma estruturação dos eventos e registos, bem como, viabilizar a correlação dos dados recebidos em tempo real.

Respeitante à gestão de rede facultada pelas plataformas FCAPS presentes, a equipa de gestão e administração de rede (SRC), apoia-se na monitorização dos dispositivos auxiliando-se em critérios de usabilidade destas ferramentas, salientando-se que estas necessitam também de inúmeras tarefas de manutenção diárias para dar a visão do estado da rede aos mais diferentes níveis funcionais (consultar Tabela 15).

Tabela 15 - Gestão diária das plataformas FCAPS da U. Porto

Aplicações	Usabilidade de gestão de redes	Gestão operacional diária
Nagios	Consulta do estado de disponibilidade das interfaces do dispositivo e dos seus serviços críticos de rede	Adicionar, remover o/ou alterar dispositivos e serviços nos ficheiros genéricos: • `hostgroups.cfg` • `hosts.cfg` • `services.cfg`
RANCID	Visualização e armazenamento das configurações dos equipamentos de rede	Adicionar ou remover equipamentos de rede, configurando os seguites diretorios: • `router.db` • `.cloginr`
RRDtool	Visualização de tráfego de dados dos equipamentos entre diferentes métricas de valores estatísticos respeitantes ao *hardware* e *software*	Adicionar ou remover equipamentos de rede aos ficheiros (`.conf`) localizados em: • `/home/netup/rrd`
Weathermap	Permite a visualização de esquemas e diagramas, demonstrando gráficos e percentagens de tráfego circulante nas interligações físicas apenas dos equipamentos de rede	Colocação de imagens alusivas à topologia no ficheiro localizado em: • `/home/netup/weathermap-php/images` Adicionar e parametrizar ficheiros com valores associados aos sistemas de rede localizados em: • `/home/netup/weathermap-php/configs`

Portal de gestão de autenticação	Gestão de contas VPN e contas wireless; Mapas de endereçamento IP; Controlo de registos dos utilizadores.	• Estruturação proprietária
Grafana	Permite consultar informação estatística, tanto ao nível dos nós de rede, bem como na vertente dos sistemas informáticos	Adição de equipamentes e terminais nos ficheiros existentes (diretoria: `/etc/telegraf`): • `balancer.conf` • `firewalls.conf` • `routers.conf` • `switchs-pólo1.conf` • `switchs-pólo2.conf` • `switchs-pólo3.conf` • `switchs-pólo4.conf` • `servers.conf` • `wifi.conf`
Cisco Prime	Visualização de informação dos sistemas *wireless*, baseando-se em gráficos estatísticos, mapas de localização geográfica dos APs, registos de utilizadores, entre outros.	• Criação de redes para serviços; • Configuração de redes de gestão e descoberta de APs; • Associação de dispositivos às controladoras; • Automatização de configurações via instanciação de templates; • Extração de relatórios específicos e personalizados às estatísticas da rede *wireless*.

Capítulo 4

4 Estudo de ferramentas de gestão de redes

4.1 Introdução

A rede de gestão da U. Porto contempla inúmeras ferramentas de gestão de redes que cobrem as cinco áreas funcionais do modelo FCAPS. Atendendo à proposta de melhoria das tecnologias utilizadas pelas plataformas de gestão de redes existentes atualmente, foram estudadas e apresentadas diferentes aplicações (capítulo 2.5), a fim de mitigar as lacunas detetadas.

Em paralelo, a própria unidade de Serviços de Redes Centrais e *Datacenters* da U. Porto, apresentou requisitos (capítulo 1.2), os quais considerou pertinentes para a melhoria dos processos de gestão operacional da rede.

Em suma, pretende-se testar as ferramentas de monitorização já existentes na U. Porto (Tabela 16), reaproveitando o ambiente de produção em que se inserem, e as plataformas Prometheus, Zabbix, Zenoss e Oxidized, as quais serão implementadas num cenário de testes.

Tabela 16 - Ferramentas atuais de gestão e monitorização de redes

Áreas Funcionais FCAPS	Plataformas de Gestão de Redes
Gestão de falhas	Nagios, Cisco Prime
Gestão de configuração	RANCID
Gestão de contabilização	Cisco Prime e Portal de gestão de autenticação (Ferramenta proprietária)
Gestão de desempenho	RRDTool, TIG Stack, Cisco Prime
Gestão de segurança	Syslog-NG, Graylog e Portal de gestão de autenticação

4.2 Ambiente de testes

Durante o planeamento do ambiente de testes tiveram de ser estudados conceitos importantes, nomeadamente, os modelos e paradigmas da gestão de redes, as respetivas metodologias, a análise da infraestrutura de rede (passiva e ativa) existente na U. Porto, bem como, o estudo e levantamento de todas as tecnologias e ferramentas inerentes ao modelo de gestão FCAPS. Entre estes conceitos, enumeram-se os princípios orientadores mais relevantes, que foram considerados para o desenvolvimento do ambiente de testes em produção:

- Compreensão da topologia da rede de gestão existente;
- Utilização do modelo de gestão FCAPS como base de estudo para mitigar lacunas, implementando assim, possíveis melhorias na gestão operacional de rede;
- Caracterização dos principais componentes e elementos de rede passíveis de serem monitorizados;
- Utilização de uma rede de gestão já existente (rede de *logs*), de forma a minimizar ao máximo, o impacto das configurações adicionais nos nós de núcleo e distribuição (maioritariamente constituída por: *firewalls*, *routers* e *switches*);
- Análise dos requisitos computacionais para implementação das plataformas de gestão de redes sugeridas;
- Recurso à solução de virtualização em operação na U. Porto (OpenStack) para instalação das ferramentas a testar (Prometheus, Zabbix, Zenoss e Oxidized.) e posterior análise comparativa entre as mesmas;

4.2.1 Análise dos requisitos computacionais

Com o objetivo de entender os requisitos computacionais para posterior implementação dos sistemas de gestão de redes, procedeu-se ao estudo e avaliação das especificações aplicacionais ao nível do *hardware* necessário, tendo por base a apreciação do poder de processamento, a capacidade de armazenamento e a memória de acesso aleatória (*Rondom Access, Memory*, RAM).

Mediante a informação adquirida e apresentada na Tabela 17, podemos aferir que os valores obtidos são genéricos, podendo sofrer alterações significativas mediante a dimensão dos nós de rede a monitorizar, bem como o respetivo número de serviços.

Tabela 17 - Requisitos computacionais genéricos das plataformas propostas

Plataforma	Número de *hosts*/serviços	Processador (cores)	Armazenamento em disco	Memoria RAM	Sistema operativo
Prometheus	Até 10000 serviços	2 CPUs	64 GB	2 GB	CentOS 7
Zabbix	Até 10000	4 CPUs	200 GB	8 GB	CentOS 7
Zenoss Core	Mais de 1000	4 CPUs	250 GB	24 GB	CentOS 7
Oxidized	Não especificado	2 CPUs	80 GB	2 GB	CentOS 7

Torna-se ainda importante salientar algumas considerações em relação aos valores pesquisados nos *sites* das empresas responsáveis pelo desenvolvimento das plataformas de gestão de redes em cima descritas (Tabela 17).

A ferramenta Prometheus não disponibiliza informação fidedigna em relação ao número mínimo de *hosts* nem serviços a monitorizar no entanto, providencia valores de referência para calcular o número de métricas guardadas por omissão num espaço temporal de quinze dias (retenção). Procedeu-se portanto, à aproximação dos valores tendo em conta as especificações técnicas previstas pela comunidade do Prometheus (versão 1.8) [63].

Em relação à plataforma Zabbix, a mesma encontra-se oficializada no que respeita aos requisitos computacionais do *hardware* recomendado (versão 4.2). O valor de armazenamento em disco (200GB) foi estimado para monitorizar até dez mil nós de rede, sendo este valor calculado com as respetivas fórmulas, de modo a possibilitar um armazenamento da informação na base de dados até um ano [64]. Destaca-se ainda, que a dimensão do espaço em disco deste sistema poderá aumentar substancialmente mediante o número de itens (serviços) a monitorizar pelo Zabbix.

Por sua vez, o Zenoss Core fornece apenas informação oficial dos requisitos computacionais baseada na sua arquitetura modular segmentada (secção 2.5.5).

Deste modo, para possibilitar a instalação de todos os componentes num só sistema informático, propõe uma solução com exigências ao nível de *hardware* um pouco superiores, necessitando de 200GB para armazenamento de informação de gestão aplicacional (*application data layer*) bem como torna-se fulcral a disponibilização de 50GB adicionais para o módulo de dados de serviços internos de controlo central (*Control Center Internal Services Data*). Adicionalmente, esta solução requer 24GB de memória RAM [65] .

Seguidamente, a ferramenta Oxidized em termos de exigências do domínio tecnológico, não requer grande poder computacional, não sendo apresentadas quaisquer informações relevantes para o seu normal desempenho, por parte da organização que o desenvolve [50]. Contudo, para o ambiente de testes, considerou-se os valores de *hardware* representados na Tabela 17, como suficientes para a implementação do Oxidized, sendo estes passíveis de serem aumentados, mediante os requisitos de rede da U. Porto.

Sistema de virtualização utilizado

A implementação das ferramentas de gestão de redes passou então, por instanciar quatro máquinas virtuais na plataforma OpenStack, segundo os requisitos genéricos estabelecidos.

Evidencia-se o sistema OpenStack, sendo este utilizado pela U. Porto para promover serviços na *cloud* aos seus utilizadores.

4.2.2 Descrição do cenário

Durante o desenvolvimento do cenário de testes, foi elaborado um estudo exaustivo da rede de gestão da U. Porto (capítulo 3.2). Isto viabilizou a conceção de um panorama realista, de forma a integrar as diferentes plataformas de gestão, permitindo assim a avaliação das funcionalidades de cada aplicação comparativamente às ferramentas já existentes, apresentadas Tabela 16.

Para o processo de integração do ambiente de testes foi utilizada a rede de gestão de *logs* da U. Porto. Esta rede possibilita a monitorização dos nós através de três principais camadas de rede, nomeadamente:

- Redes de serviços IP (encaminhamento de dados);
- Redes de serviços IP/MPLS;

- Redes de gestão IP/MPLS.

A rede de gestão de *logs*, tem acessos autorizados nas seis *firewalls* onde residem as redes de dispositivos passíveis de monitorizar. Do ponto de vista da *firewall* de gestão, podemos confirmar que a rede administrativa local (gestão física) e a rede administrativa externa (gestão remota), providenciam configurações ao nível da camada de transporte (ACLs) e da camada de rede (rotas estáticas) do modelo TCP/IP, de modo a alcançar qualquer rede presente na U. Porto.

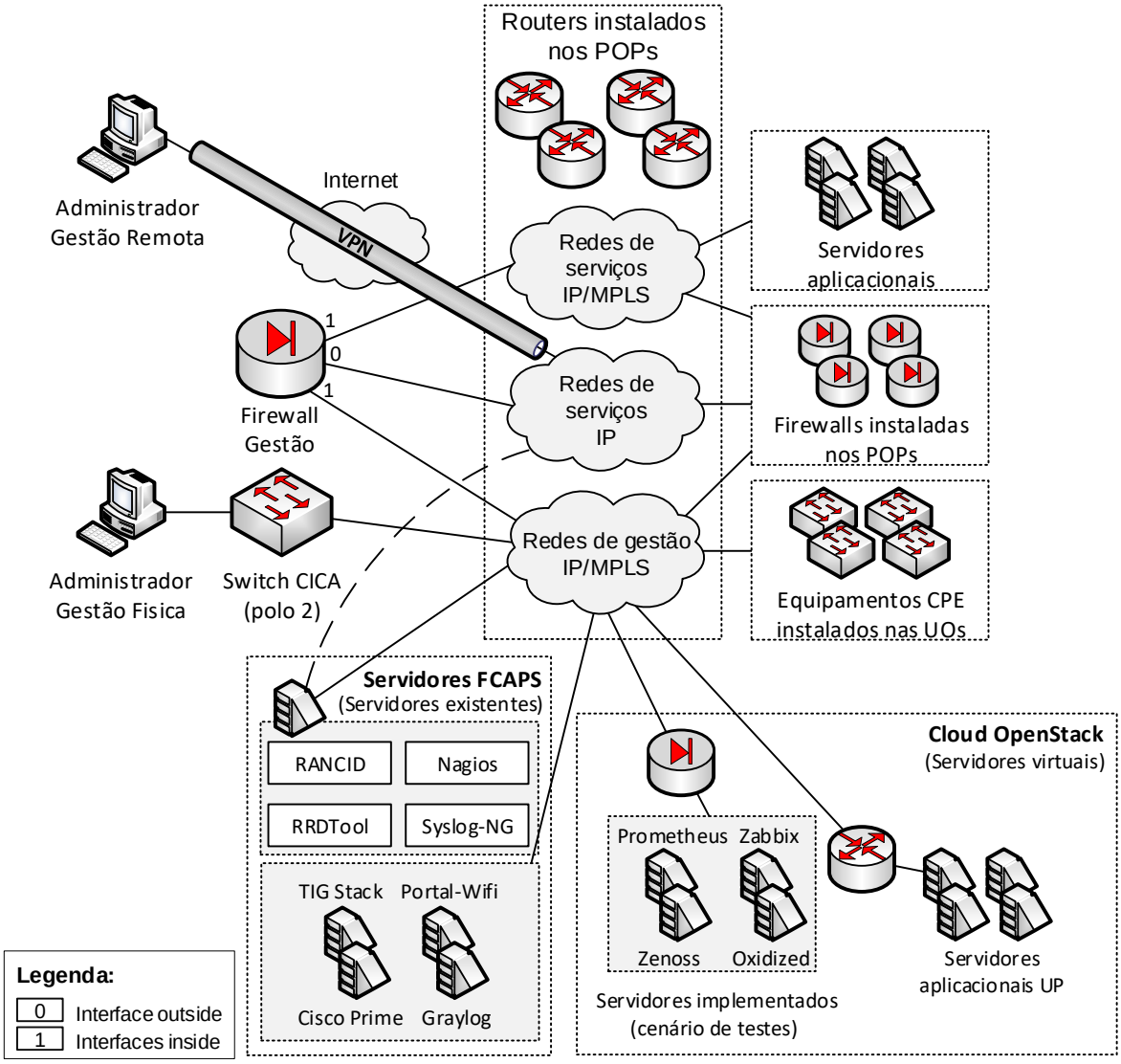


Figura 34 - Esquematização da infraestrutura para integração das ferramentas FCAPS

Após validação das camadas de rede e transporte, foi confirmada a camada de ligação de dados. Para que o domínio *broadcast* pertencente à rede de gestão de *logs* seja transportado até aos equipamentos agregadores, CPEs, *firewalls*, entre outros, analisou-se o respetivo serviço VPLS instanciado e configurado nos *routers* presentes nos POPs. Cada dispositivo de rede interligado a um *attachment circuit*, contém então, a VLANs inerente à rede gestão de equipamentos.

As restantes redes de gestão e de serviços seguem o mesmo conceito, sendo alcançáves por qualquer endereço IP da rede de gestão de *logs*. Deste modo, os servidores do cenário de testes conseguem ter comunicação com os diferentes elementos de rede (Figura 34). Ainda assim, para que seja possível monitorizar os dispositivos de rede, foi imprescindível configurar o acesso aos endereços IP de origem de cada servidor implementado (Prometheus, Zabbix e Zenoss) nos nós de rede correspondentes, autorizando deste modo, a utilização do protocolo SNMP e respetiva porta.

No que respeita à ferramenta de gestão de configurações, Oxidized, esta necessitou de acessos aos diferentes componentes de rede via SSH e Telnet, sendo estes configurados em todos os dispositivos propostos, de modo a descarregar as configurações (*backups*).

Em relação às plataformas de gestão de redes existentes, que cobrem as diferentes áreas do modelo FCAPS, as mesmas interagem com as diferentes camadas de redes ilustradas na Figura 34. Diferencialmente dos servidores existentes, os novos servidores implementados atendem à melhoria das áreas da gestão funcional dos principais elementos de rede apresentados:

- *Routers*;
- *Switches* agregadores;
- *Switches* CPEs;
- *Firewalls* de núcleo e distribuição;
- Servidores aplicacionais de serviços críticos de rede (DHCP, DNS, entre outros).

A topologia de funcionamento lógico do cenário de monitorização teve por base o modelo arquitetónico de rede da U. Porto análogo à Figura 34. Tendo em conta a enorme extensão desta topologia, optou-se por monitorizar os parâmetros mais críticos dos equipamentos de rede:

- Utilização de CPU;

- Utilização de memória RAM;
- Débito das interfaces;
- Disponibilidade do elemento de rede.

Por outro lado, na classe dos sistemas computacionais foram destacados alguns critérios que permitem verificar a utilização do *hardware* e do *software* dos servidores aplicacionais. Entre estes parâmetros enumeram-se os seguintes:

- Utilização de CPU;
- Utilização de memória RAM;
- Utilização de discos/partições;
- Estado dos serviços aplicacionais;
- Disponibilidade do sistema informático.

4.3 Critérios de seleção e implementação das plataformas

Com o desenvolvimento deste subcapítulo pretende-se descrever as funcionalidades preponderantes de todas as plataformas estudadas, de acordo com a análise de experimentação, objetivando as mais e menos valias de cada ferramenta de gestão de redes.

Os critérios de seleção das ferramentas de gestão de redes são completamente dependentes de inúmeros fatores. De facto, a análise entre as ferramentas de gestão de rede, implicou um maior aprofundamento, nomeadamente ao nível teórico-prático, traduzindo-se na configuração, na monitorização de dispositivos e no tratamento da gestão de eventos e alertas. Entre muitos outros critérios, os experienciados serão a base de fundamentação da nossa opção.

Cisco Prime e Portal de gestão de autenticação

Primeiramente, é relevante salientar que as plataformas Cisco Prime e Portal-de gestão de autenticação não irão ser comparadas com outras ferramentas propostas. Estas aplicações revelam-se muito importantes cumprindo de forma eficaz as suas tarefas inerentes ao modelo FCAPS.

O Cisco Prime garante a monitorização da rede sem fios com recurso à gestão das controladoras *wireless*. Neste ponto, o Cisco Prime consegue satisfazer a sua tarefa eficientemente, possibilitando assim, a gestão de configuração e alarmística dos APs, bem como demonstra a informação de desempenho através de gráficos superintuitivos.

A gestão de contabilização fornece através de indicadores, como por exemplo, o registo do número de utilizadores que estão autenticados na rede sem fios. Detalha bastante a demanda de rede sem fios, viabilizando a disponibilidade de relatórios estatísticos muito concisos e pormenorizados.

Em relação à aplicação Portal de gestão de autenticação, destaca-se a sua enorme mais-valia no processo da gestão operacional da rede *wireless*. Na verdade, esta plataforma foi desenvolvida pela equipa de serviços de rede centrais da U. Porto com o intuito de gerir os acessos VPN.

O Portal de gestão de autenticação descreve as diferentes tarefas operacionais diárias ao nível da rede *wireless*, promovendo a criação de contas VPN (como intermediário através da instanciação de dados inerentes aos protocolos AAA), a correlação de *logs*, o mapeamento de redes IP com os diferentes elementos de rede, contemplando ainda outras funcionalidades.

Revela-se uma ferramenta bastante empírica, de fácil utilização e que não carece de entrar no modelo comparativo subsequente. No entanto, é de realçar que a base de dados desta plataforma, não congrega o modelo de alta disponibilidade, sendo incapaz de ser tolerante a falhas devido à falta de redundância.

Nagios

Como ferramenta de gestão alarmística em produção na rede de dados da U. Porto, o Nagios demonstrou-se bastante modelar e personalizável. Por outro lado, o seu modo de configuração revela-se um pouco complexo e manual, podendo até tornar-se repetitivo pelo elevado número de *hosts* que são diariamente adicionados, removidos ou alterados pela equipa de operação das redes de comunicação de dados.

A configuração de serviços pode ser bastante morosa, nomeadamente quando é necessária a monitorização de um elevado número de parâmetros ou serviços, em operação num servidor. Para além disso, esta situação é agravada pelo facto do Nagios não possuir suporte nativo para o protocolo SNMP, recorrendo a *scripts* externos que representam uma dependência adicional ao sistema, demonstrando a ineficiência na recolha de informação.

Um dos principais critérios de uma rede da dimensão da U. Porto é a alta disponibilidade dos serviços, assim sendo, o Nagios cumpre este princípio com a redundância de sistemas, recorrendo-se a um *script* manual que é utilizado por parte dos administradores de rede para sincronizar as configurações, quando estas sofrem alterações. Isto é um recurso demasiado estático que pode acarretar inconsistências na configuração, sendo uma ação passível de erro humano.

Analisando o seu perfil alarmístico, constatou-se que por vezes as mensagens de *email* são pouco imediatas, apresentando ainda a deficiência de possibilitar a definição de diferentes níveis de criticidade. Ou seja, não é possível tratar os alertas dos equipamentos tendo em conta os diferentes impactos que estes representam para a rede.

No que respeita ao interface gráfico, a visualização da informação é pouco apelativa, não existindo a possibilidade de fazer qualquer tipo de operação ao nível de configurações. Os resultados da monitorização são apenas visíveis sob a forma de estados, indisponibilizando assim, os valores de desempenho recolhidos (criação de *dashboards*), tornando impossível a análise de tendências dos elementos monitorizados.

Prometheus

A plataforma de gestão de redes Prometheus providenciou um conceito diferente de monitorizar os dispositivos de rede. Embora a sua interface gráfica deixe um pouco a desejar por não conter funcionalidades de usabilidade ("*user friendly"*), a mesma possibilita a utilização de comandos em formato de *query*, para descobrir múltiplas informações de desempenho dos *hosts* monitorizados.

As suas potencialidades ao nível de desempenho são bastante preponderantes e detalhadas, não recorrendo a muita capacidade computacional.

A base de dados demostrou-se muito resiliente, sem qualquer tipo de erro ou falha aparente, agregando a informação num formato compacto. No entanto, ao nível da gestão de falhas, exige a agregação de um módulo de alertas que dispõe de poucas aplicabilidades, apresentando problemas semelhantes aos enumerados na secção dedicada à análise do Nagios.

Em relação aos comandos utilizados para extrair a informação dos equipamentos, estes revelaram-se complexos, exigindo um elevado processo de autoaprendizagem. Neste ponto, salienta-se como uma aplicação inadequável no que respeita à gestão operacional por parte da equipa de operação, obrigando à existência de sessões de formação exaustivas, para que seja possível tirar partido das reais capacidades e funcionalidades desta plataforma.

Paralelamente a arquitetura modelar do Prometheus impõe um forte conhecimento ao nível de programação. Ao mesmo tempo, quando devidamente parametrizado poderá fazer mais sentido para os novos ambientes de rede definidos por *software* (*Software Defined Network*, SDN), por este não depender da camada de rede para realizar a autodescoberta de serviços.

Zenoss Core

Durante o processo de instalação e configuração do Zenoss, foram sentidas diversas dificuldades. Este facto, deve-se essencialmente à falta de documentação oficial, por parte da comunidade, para as novas versões *open source* disponíveis.

Em contrapartida, esta ferramenta disponibiliza a monitorização de desempenho e de disponibilidade de recursos. Apesar disso, foi observado que o método de operação do Zenoss é otimizado para modelos de monitorização sem utilização de agentes. Isto poderá ser uma menos valia mediante a topologia de redes de comunicação e sistemas informáticos em operação na Universidade.

Ao nível da visualização e configuração, a interface *web* do Zenoss, proporciona uma parametrização de alto nível de objetos, garantindo também a gestão de eventos, de forma compreensível e atrativa. Independentemente de ser uma interface apelativa, apresenta alguma demora na apresentação da informação e, certas configurações podem ser complexas, tornando algumas funcionalidades, mais particulares, pouco empíricas.

A construção de *templates* não é complicada, mas requer conhecimentos de estruturação da aplicação e dos métodos a adotar na monitorização de dispositivos. Sob outra perspetiva, as operações de gestão de equipamentos são mais acessíveis e intuitivas

Zabbix

O Zabbix foi a ferramenta que conseguiu, da melhor forma, agregar as funcionalidades de alarmística, propostas pelo Nagios, com as funcionalidades de estatística, oferecidas pela ferramenta RRDTool, existente atualmente e em operação na U. Porto. Ao nível operacional, o mesmo é otimizado para a utilização de agente, sendo passível de um maior entendimento para os restantes métodos de monitorização.

A interface *web* disponibiliza diferentes formas de configurar a aplicação, bem como os itens (serviços) a monitorizar num dado *host*. Visualmente pode expor demasiada informação num primeiro contacto, tornando-se pouco clara em relação aos sistemas monitorizados.

A celeridade com que múltiplos elementos de rede são monitorizados é a sua grande

vantagem. Isto deve-se ao conceito de *template* que o Zabbix preconiza, ao contrário da plataforma Nagios. Além disso, a automatização de processos, com recurso ao mecanismo de autodescoberta, revelou-se um ponto de destaque, tanto na perceção dos dispositivos de rede, quanto aos serviços a monitorizar.

Os alertas são extremamente simples de configurar, oferecendo grande detalhe, suportando ainda a execução e várias ações que permitem correlacionar eventos, tais como o envio de um *email* ou SMS (*Short Message Service*), ou notificação via *Telegram* (aplicação) que tem como vantagem a disponibilização da informação de desempenho (gráfico estatístico).

RRDTool

Durante a análise de experimentação da ferramenta RRDTool presente atualmente na rede de comunicações, caracterizou-se alguns riscos e lacunas.

Primeiramente esta ferramenta foi agregada a uma aplicação de visualização denominada Weathermap. Em termos de configuração, a correlação entre ferramentas, impõe um maior risco de falha no processo de gestão operacional diário da rede. Para além disso, o suporte e desenvolvimento por parte do Weathermap não se encontra atualmente em vigor.

Paralelamente, os dados históricos armazenados pelo RRDTool não evidenciam valores suficientes. O processo de configuração da aplicação RRDTool é muito manual, revelando a falta de flexibilidade na medição de valores inerentes ao desempenho.

Esta ferramenta demonstrou também a impossibilidade de criação de *dashbords*, obrigando ao uso do Weathermap para esse efeito. No entanto a visualização desses dados não é nada apelativa por parte do Weathermap, obrigando a conhecimentos avançados de HTML (*HyperText Markup Language*) e PHP, para estruturar um bom *layout* de gráficos.

TIG Stack

Atualmente a U. Porto dispõe de um conjunto de ferramentas, nomeadamente, o Telegraf, o InfluxDB e o Grafana implementadas, para colmatar as lacunas principais do sistema RRDTool. Contudo, tanto o TIG Stack quanto o RRDTool encontram-se em produção devido à apreciação das funcionalidades e análise comparativa entre ferramentas, não ter sido concluída.

Embora o RRDTool seja baseado em configuração manual, o TIG Stack acaba por também o ser. A sua composição alia três ferramentas e a gestão operacional acaba por ganhar alguma complexidade, nomeadamente no processo de adicionar um equipamento e criar gráficos para

visualizar a sua informação de desempenho. Quando é necessário remover um equipamento, a tarefa exige conhecimentos muito específicos de InfluxDB, a fim de se remover as respetivas entradas na base de dados.

O TIG Stack acaba por ter um funcionamento particular que obriga a operações bastante estáticas e repetitivas como acontece com o Nagios. Maioritariamente as suas configurações ocorrem no Telegraf, sendo que estas são passíveis de erro humano, pois as suas parametrizações só podem ser realizadas por linha de comandos.

No que diz respeito às tarefas de *troubleshooting* do TIG Stack, foi adicionada mais uma aplicação, denominada Chronograf, essencial para se executar a manutenção da base de dados e correlacionar possíveis causas de erro. Isto por um lado auxilia na manutenção aplicacional, através de ambiente gráfico, promovendo a resolução de possíveis complicações. Ainda assim, obriga à gestão de mais uma aplicação por parte da equipa de suporte, causando maior entropia no processo da gestão estatística.

Em relação ao RRDTool, o TIG Stack ganha mais relevo pela quantidade de informação que é armazenada na base de dados (InfluxDB). No entanto, o Zabbix disponibiliza, de igual modo, a informação de desempenho, e integra expressões regulares através da sua API, facto que providencia a automatização e visualização dos dados com o Grafana. A adoção do TIG Stack foi uma tentativa de tentar colmatar a falta de informação de desempenho par parte do Nagios e, neste sentido, o Zabbix consegue também solucionar esta questão.

A utilização do Zabbix aliada com o Grafana poderia ser uma perspetiva mais interessante concentrando as tarefas de gestão operacional da rede em apenas duas plataformas, integrando, deste modo, a gestão de falhas e a gestão de desempenho.

RANCID e Oxidized

O RANCID é a plataforma de gestão de configurações que se encontra em produção na U. Porto. Esta ferramenta apresenta uma configuração custosa e pouco flexível. A carência da interface gráfica acarreta uma enorme dificuldade em verificar o estado das configurações.

Um dos principais pontos experienciados e determinados como negativos, é a sua incompatibilidade com novos equipamentos e respetivas versões de *software*, obrigando assim, a recorrer sistematicamente a *scripts* manuais.

Em contrapartida, o Oxidized não só resolve as carências anteriormente citadas, como permite a visualização das configurações armazenadas através de uma interface gráfica,

determinando cada linha de código alterada mediante a diferença detetada na configuração de um dado equipamento monitorizado.

A configuração e instalação do Oxidized é extremamente rápida e intuitiva, não tendo sido verificados requisitos computacionais de dimensão superior ao RANCID. A estrutura em termos de diretorias é semelhante, conseguindo, ainda assim, ser de melhor entendimento. Outro ponto positivo a favor do Oxidized é a possibilidade que dispõe em relação à migração dos ficheiros de configuração da ferramenta RANCID (`router.db`), isto é, através da interface *web* é possível inserir o conjunto de equipamentos monitorizados pelo RANCID, facto que facilita a tarefa de migração de plataformas.

Syslog-NG e Graylog

Embora o Syslog-NG ainda se encontre em produção, o Graylog foi recentemente adotado para colmatar algumas limitações ao nível de gestão de segurança. Relativamente à gestão de mensagens de *log* recolhidas pelo Syslog-NG, foi identificado que as técnicas de recolha, processamento, armazenamento e análise, eram pouco eficientes. Concretamente, estas mensagens de *log* não eram filtradas devidamente nem armazenadas também em ficheiros de texto.

O Graylog foi então inicialmente configurado em ambiente de testes pela unidade SRC, à qual o autor do presente projeto pertence, permitindo assim a compreensão do funcionamento desta solução e as mais-valias inerentes a esta plataforma. Destaca-se como fator diferenciador do Syslog-NG, a simplicidade que o Graylog concerne, não só na recolha de mensagens de *log*, mas também na possibilidade de interação com a ferramenta, visto ser possível efetuar todas as configurações, de forma centralizada e através de interface gráfica.

A arquitetura do Graylog possibilitou a implementação do conceito de alta disponibilidade, garantindo o princípio da resiliência, e a melhoria do desempenho no processamento de *logs*, sendo que o Syslog-NG não fornece esta característica de enorme necessidade, face à dimensão do número de equipamentos constituintes que compõe a rede de comunicação e os sistemas em operação da U. Porto.

4.4 Análise comparativa e resultados

Na secção 4.3 foi realizada uma apresentação que integrou um conjunto de critérios baseados no estudo e implementação das ferramentas propostas para a gestão de redes e sistemas.

O objetivo deste subcapítulo é especificar esses critérios, de forma a comparar as ferramentas de gestão e monitorização com base no modelo FCAPS.

Para a demonstração de resultados obtidos nas comparações e para melhor compreensão, serão apresentadas diferentes tabelas com classificações numéricas. Para esse efeito, utilizou-se a numeração de 1 a 5, sendo o valor 1 o mínimo a ser atribuído e o valor 5 o mais elevado.

Tabela 18 - Apreciação das ferramentas de gestão de falhas

Gestão de falhas				
Critérios / Ferramentas	Nagios	Prometheus	Zenoss Core	Zabbix
Documentação	5	3	3	5
Escalabilidade	4	3	3	4
Autodescoberta de dispositivos	1	3	5	5
Interface gráfica para configurações	2	1	4	4
Arquitetura modelar para alta disponibilidade e resiliência	1	5	5	5
Gestão operacional	2	1	4	4
Total	**15**	**16**	**24**	**27**

Mediante os resultados da Tabela 18, torna-se importante evidenciar as melhores características funcionais de cada aplicação. O Nagios e o Zabbix disponibilizam mais documentação fidedigna que as restantes plataformas. Embora o Nagios necessite da integração e de um elevado número de *plugins*, consegue, ainda assim, destacar-se como uma plataforma escalável ao mesmo nível do Zabbix.

Por outro lado, o Zenoss e o Zabbix providenciam a autodescoberta dos dispositivos de rede, sendo uma vantagem significativa em relação ao Nagios e o Prometheus. Ao nível da interface gráfica tanto o Zenoss como o Zabbix fornecem a possibilidade de parametrizar toda a informação de gestão e alarmística, conseguindo assim monitorizar os equipamentos de forma mais eficiente.

O Prometheus contém uma arquitetura modelar, no entanto a documentação é apenas disponibilizada no *site* oficial, e não é orientado à descoberta de dispositivos, embora possa fazê-lo.

Em relação ao critério de gestão operacional, avaliou-se a interação das ferramentas no ambiente de testes, tendo em conta a quantidade de configurações diárias passivas de serem realizadas, e a automatização de processos que as plataformas proporcionam. A este nível, o Zenoss e o Zabbix comportam as tarefas de gestão de falhas diárias da U. Porto, com maior destaque.

A diferenciação entre o Zenoss e o Zabbix não é muito significativa, no entanto o Zabbix mostra-se superior pela demanda existente no que respeita à documentação. Com base na análise efetuada às aplicações de gestão de falhas, chegou-se à conclusão de que o Zabbix é a ferramenta que melhor se adequa aos requisitos definidos para a rede da U. Porto.

Tabela 19 - Apreciação das ferramentas de gestão desempenho

Gestão de desempenho					
Ferramentas / Critérios	Prometheus	Zenoss Core	Zabbix	RRDTool / Weathermap	TIG Stack
Documentação	3	3	5	2	4
Criação flexível de dashbords	2	4	4	1	3
Autodescoberta de serviços	5	4	4	1	1
Interface gráfica	2	4	4	1	4
Gestão operacional	2	4	4	2	2
Total	**14**	**19**	**21**	**7**	**14**

Após o estudo teórico-prático das ferramentas de desempenho foram destacados os critérios mais orientados às necessidades experienciadas por parte da infraestrutura de rede e sistemas da U. Porto.

Como se pode observar na Tabela 19, o Zabbix volta a ter relevo em relação às restantes aplicações. De facto, o TIG Stack carece dos mesmos problemas ao nível operacional que o Nagios e do RRDTool, nomeadamente, a edição repetitiva de ficheiros. Em paralelo, a ideia passa por diminuir o número de plataformas, a fim de otimizar e centralizar as tarefas sistemáticas.

O Zabbix mostra-se novamente superior face às restantes aplicações, permitindo assim, utilizar a mesma plataforma para a gestão alarmística e a gestão estatística, bem como, esta conjugação alia a correlação entre os dados obtidos com o objetivo de mitigar possíveis causas na eventual falha ou mau funcionamento de equipamentos.

Tabela 20 - Apreciação das ferramentas de gestão de configuração

Gestão de configuração		
Critérios / Ferramentas	RANCID	Oxidized
Configuração flexível	3	4
Compatibilidade com diferentes modelos/fabricantes	2	4
Interface gráfica	1	4
Gestão operacional	2	4
Total	**8**	**16**

Relativamente às ferramentas de gestão de configuração, o Oxidized demonstrou-se superior em relação ao RANCID (Tabela 20). A flexibilidade de configuração, em termos de diretorias não é o fator determinante. Porém, a compatibilidade entre modelos e respetivas marcas, faz do Oxidized uma aplicação diferenciada perante o RANCID, garantindo uma maior escalabilidade e a adição de novos equipamentos de rede sem correr riscos de incompatibilidades.

Ao grau da gestão operacional, a interface *web* disponibilizada pelo Oxidized, promove uma maior eficiência na consulta das configurações armazenadas, para além de ser apresentado visualmente todas as alterações realizadas na configuração.

Tabela 21 - Apreciação das ferramentas de gestão de segurança

Gestão de segurança		
Critérios / Ferramentas	Syslog-NG	Graylog
Capacidade de recolha, armazenamento e análise de *logs*	2	4
Arquitetura modelar para alta disponibilidade e resiliência	1	5
Interface gráfica	1	4
Gestão operacional	2	4
Total	**6**	**17**

De modo a promover uma gestão de segurança eficiente foram comparadas as ferramentas Syslog-NG e Graylog, as quais já se encontram implementadas na U. Porto.

Atendendo à informação apresentada na Tabela 21, fica percetível a melhoria considerável em manter o Graylog e descontinuar o Syslog-NG. De facto, o Syslog-NG não apresenta a capacidade de processamento, recolha, armazenamento e análise de *logs*, que o Graylog oferece.

O Graylog contempla um modelo arquitetónico que proporciona a alta disponibilidade, salvaguardando a instituição da questão essencial, nomeadamente a perda de *logs*. Assim sendo, o Syslog-NG não representa benefícios para a rede da U. Porto, visto que o Graylog já se encontra em produção.

Capítulo 5

5 Implementação do sistema de monitorização centralizado

5.1 Introdução

O objetivo fundamental da proposta para a renovação do sistema de monitorização alarmística e estatística da Universidade do Porto, advém do crescimento exponencial dos seus sistemas e equipamentos de rede, os quais geram uma necessidade superior no que concerne à supervisão de toda a infraestrutura ativa.

Os sistemas de monitorização em atual funcionamento (Nagios e RRDTool), embora cumpram efetivamente com a maioria dos requisitos, não oferecem a simplicidade na sua gestão diária, nem providenciam as potencialidades de uma ferramenta mais moderna.

As novas ferramentas recorrem a modelos e paradigmas de base de dados relacionais, criando assim, a possibilidade de correlacionar diferentes métricas através de *dashbords*, produzindo também, uma gestão e administração de sistemas mais efetiva, dinâmica e menos morosa no que respeita à adição de *hosts* e serviços a monitorizar.

Deste modo, perante as principais ferramentas estudadas, a proposta para a aplicação adotada recai sobre o sistema de monitorização Zabbix, sendo este escolhido com base nos critérios estabelecidos no estudo da seleção de ferramentas (Tabela 18 e Tabela 19).

5.2 Componentes integrantes

Nativamente, a arquitetura disponibilizada pela comunidade do Zabbix não é orientada à segregação dos componentes aplicacionais. No entanto, com um estudo exaustivo, compreendeu-se que a melhor opção para desenhar uma arquitetura aplicacional altamente funcional, deve não só dissociar cada componente pelo aumento intrínseco do desempenho, mas também, possibilitar mecanismos de resiliência através da duplicação dos servidores integrantes em cada *cluster*.

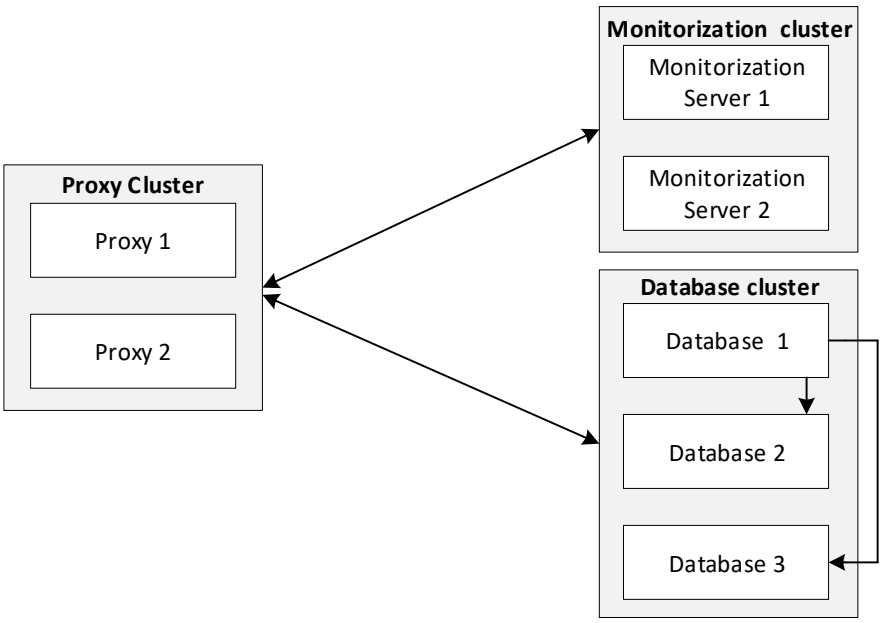


Figura 35 - Componentes da solução de monitorização

Num ponto de vista global, a arquitetura aplicacional sugerida deve compreender os três elementos que se seguem, pela seguinte ordem de implementação:

1. Base de dados;
2. *proxys*;
3. Servidores de monitorização.

5.3 Especificações das aplicações

A conceção da arquitetura para o modelo de monitorização sugerido, levou a cabo um processo de aprendizagem baseado numa pesquisa intensiva sobre as principais ferramentas *open-source* que o mercado disponibiliza. Os modelos de armazenamento estudados baseiam-se na linguagem de consulta de base de dados MySQL e PostgreSQL [66] (Anexo IX).

A escolha deste mecanismo de gestão de armazenamento prende-se essencialmente na *performance* que cada linguagem pode oferecer, garantindo de igual modo a integridade dos dados. Como modelos de bases de dados que dependem de transações diretas, o MySQL revela-se superior [67]. Por outro lado, tendo em conta os modelos de base de dados de maior dimensão, o PostgreSQL consegue corresponder melhor às espectativas dos sistemas que dependem de consultas mais complexas [67].

A implementação da arquitetura de base de dados recorreu sempre ao método de encaminhamento de dados via *software* HAproxy, de modo a criar um mecanismo de distribuição de carga altamente resiliente e independente dos nós de armazenamento.

Tabela 22 - Aplicações de base de dados testadas

Componentes computacionais de armazenamento de dados		**Aplicações**	**Versão**
Base de dados	PostgreSQL	postgresql12-devel	12.1-2PGDG.rhel8
		citus_12	9.0.1-1.rhel8
	MySQL	mariadb-server	10.4.10.el7
		Percona-XtraDB-cluster-57	5.7.27-31.39.1.el7
proxys	PostgreSQL	pgbouncer	1.12.0-1.rhel7
	MySQL	proxysql2	2.0.7-1.2.el7
		mariadb-client	10.4.10.el7

Com auxílio da Tabela 22, é possível observar todas as aplicações da componente de armazenamento de dados que foram experimentadas durante o desenvolvimento da solução.

Concretamente, o PostgreSQL foi configurado em sistemas virtualizados numa infraestrutura de serviços em *cloud*, sendo que a arquitetura Citus [68] é a mais adequada a esse cenário. Esta arquitetura é orientada ao aumento exponencial do número de nós para armazenamento de dados, face às necessidades de escalabilidade do *cluster*.

Visto a solução de monitorização não poder ficar em produção no sistema de virtualização do âmbito *cloud* (OpenStack) da U. Porto, a arquitetura Citus não é a mais viável. Como tal, dentro do paradigma da linguagem MySQL, foram testadas diferentes derivantes *open-source* do MySQL proprietário da Oracle [69].

Primeiramente, toda a solução foi testada com o mecanismo de gestão de armazenamento MariaDB. Derivado a diferentes problemas, foi detetado que os nós de armazenamento deixavam de estar sincronizados, excedendo o número de ligações permitido (2000 ligações). O consumo de memória RAM era muito elevado, sendo que 16GB (ver Tabela 24) não se mostrou suficiente. Depois de um conjunto extenso de testes detetou-se uma incompatibilidade com o servidor Zabbix [70].

O facto de o MariaDB obrigar ao uso de um *proxy* para a construção de um *cluster* de base de dados, sem recorrer a modelos "pagos" (MaxScale), dificultou todo o *troubleshooting* realizado. O HAproxy, não recorre à camada aplicacional do modelo TCP/IP para analisar o cabeçalho, comutando assim os pacotes IP recebidos com base na inspeção da camada de transporte e de rede. Como tal, tornou-se ineficaz detetar o problema desta forma.

Contudo, foram investigadas novas formas de solucionar a problemática referida. Compreendeu-se que o Percona tinha inúmeros benefícios face ao MariaDB sendo a arquitetura de base de dados mais idêntica ao MySQL proprietário [69] (Anexo X).

O modelo preconizado pelo Percona contém encaminhadores baseados em proxySQL que fazem a retrospeção dos cabeçalhos da camada aplicacional, possibilitando assim, a descoberta de possíveis causas de falha nos sistemas que decompõe o *cluster* de base de dados. No processo inicial de configuração do Percona foi detetado o erro (`ERROR 1105 (HY000)`), o qual demonstra a inexistência de chaves primárias (*primary-key*) para identificar exclusivamente cada registo de uma tabela (*scritpt* de criação de base de dados do Zabbix), sendo este o mesmo motivo que gerou o impacto na falha do MariaDB (Figura 96) Após esse ajuste, o Percona não teve mais falhas críticas, sofrendo apenas *upgrades* de *performance* no motor de base de dados do MySQL, mais concretamente o InnoDB (Figura 85).

Tabela 23 - Aplicações da solução proposta

Componentes Computacionais	Aplicações instaladas	Versões das aplicações
Base de dados	Percona-XtraDB-cluster-57	5.7.27-31.39.1.el7
proxys	haproxy	2.0.7-1.el7
	proxysql2	2.0.7-1.2.el7
	zabbix-proxy-sqlite3	4.4.3-1.el7
	pacemaker	1.1.20-5.el7_7.2
Servidores de monitorização	zabbix-server-mysql	4.4.3-1.el7
	zabbix-web-mysql	4.4.3-1.el7
	pacemaker	1.1.20-5.el7_7.2

A solução passou então por diversas fases de configuração e testes que serão descritas ao longo deste relatório técnico. Enumeram-se na Tabela 23 as aplicações que efetivamente se mostraram eficientes no domínio das componentes computacionais, as quais foram testadas em ambiente de produção na plataforma KVM (K*ernel Virtual Machine*) da U. Porto.

A estrutura resiliente da arquitetura implementada, garante que um dos nós de computação possa falhar dentro de cada componente, com a exceção do *cluster* de armazenamento, visto ser constituído por três servidores, podendo este permanecer em funcionamento apenas com um nó. Foram testadas duas aplicações distintas, o Pacemaker e o Keepalive. Ambas utilizam o conceito de endereço VIP (*Virtual IP Address*) para que apenas a máquina primária responda ao endereço virtual.

Com diferentes testes para forçar falhas propositadas, compreendeu-se que o Keepalive não consegue garantir a redundância com base nos processos da máquina, ou seja, se um dos processos parar, a sistema continua a responder ao endereço VIP. Outro aspeto muito importante são as prioridades dos processos aplicacionais, deste modo, o servidor Zabbix precisa de iniciar ou terminar os seus processos por uma dada ordem, para que a máquina não fique indisponível. O Pacemaker recorre então às configurações do Corosync para poder fazer essa gestão dos processos, por sua vez o Keepalive valida os estados de conetividade da rede com recurso ao protocolo VRRP (*Virtual Router Redundancy Protocol*). O entendimento do funcionamento destas duas aplicações foi preponderante para definir uma uniformização no modelo de resiliência, recorrendo ao Pacemaker para possibilitar mecanismos de redundância entre os *proxys* e os servidores de monitorização Zabbix.

Os sistemas de armazenamento, como referido anteriormente, recorrem ao Percona para disponibilizar o modelo MySQL, com a particularidade que o Percona disponibiliza um método de sincronismo entre os nós, baseado em Galera XtraDB *cluster*, de forma a validar possíveis falhas ou inconsistências dos dados. Originalmente, o Percona traz um mecanismo denominado por Xtrabackup, que resolve os problemas de indisponibilidade dos sistemas associados ao *cluster* de armazenamento, sem tornar o nó que transmite a informação indisponível.

5.4 Requisitos funcionais

Depois da análise exaustiva das componentes aplicacionais, avaliaram-se os requisitos computacionais, de modo a integrar a solução de monitorização num dos ambientes de virtualização da U. Porto. Na Tabela 24 enumeram-se as especificações de *hardware* que cada sistema computacional deve ter, tendo em conta os requisitos de monitorização dos equipamentos ativos da rede da U. Porto (consultar Tabela 11).

Tabela 24 - Requisitos funcionais da solução de monitorização

Componentes Computacionais	Processador (cores)	Memória RAM	Armazenamento em disco	Sistema operativo
Zabbix *cluster*				
Servidor zabbix 1	4 CPUs	8 GB	80 GB	CentOS 7
Servidor zabbix 2	4 CPUs	8 GB	80 GB	CentOS 7
Percona XtraDB *cluster*				
Percona DB 1	4 CPUs	16 GB	300 GB	CentOS 7
Percona DB 2	4 CPUs	16 GB	300 GB	CentOS 7
Percona DB 3	4 CPUs	16 GB	300 GB	CentOS 7
proxy cluster				
proxy 1	2 CPUs	4 GB	40 GB	CentOS 7
proxy 2	2 CPUs	4 GB	40 GB	CentOS 7

Os valores para a componente de monitorização foram calculados mediante a informação disponibilizada pelo Zabbix SIA [64] . Tendo em conta que arquitetura não é só composta pelos servidores Zabbix, foram ainda avaliados os requisitos funcionais dos restantes componentes integrantes, tais como, os sistemas de base de dados e os servidores de *proxy*.

Relativamente aos nós de armazenamento, foi calculado um espaço total em disco, de modo a garantir os dados recolhidos até um ano [71]. Este valor perfaz um total de 300GB por cada nó de computação, lembrando que todos os nós que decompõe o *cluster* guardam a mesma informação (modelo multi-mestre) e, portanto, o *cluster* tem uma capacidade máxima igual ao espaço de armazenamento de um nó.

Os requisitos de *hardware*, especialmente ao nível da unidade central de processamento (CPU), e a memória de acesso aleatório (RAM) dos sistemas de armazenamento, foram determinados e ajustados, tendo em conta os valores mínimos aconselhados pela documentação oficial do Percona-XtraDB-cluster [72].

Por outra perspetiva, os *proxys* não requerem grande poder computacional, visto serem apenas intermediários para o encaminhamento de dados, entre o *cluster* de Zabbix, o *cluster* de base de dados e os dispositivos monitorizados.

Procedeu-se à utilização das especificações recomendadas para os dois sistemas HAproxy de *High-level*, proporcionando, deste modo, mais de 20000 ligações em simultâneo [73]. Torna-se ainda importante evidenciar que este sistema irá suportar também os *softwares* zabbix-proxy-sqlite3 e proxysql2 contudo, as especificações referenciadas pelo HAproxy garantem de igual modo o funcionamento eficiente destas aplicações.

Cada um dos componentes computacionais integrantes na arquitetura aplicacional foi munido com o sistema operativo CentOS 7 (7.7.1908 *core*) de 64 bits. Todos estes sistemas utilizam o modelo de virtualização integrado na plataforma KVM da U. Porto, o que possibilita qualquer *upgrade* ao nível de *hardware*, mais concretamente, o aumento de CPUs e o incremento de mais memória RAM ou ainda, a ampliação do espaço em disco. Isto resolve quaisquer carências ao nível de escalabilidade que possam surgir e é sem dúvida uma grande benesse na aquisição dos sistemas virtualizados.

5.5 Funcionamento genérico

O entendimento do funcionamento genérico entre as componentes integrantes da solução de monotorização proposta é essencial para a compreensão das configurações dos capítulos seguintes.

O esquema de endereçamento IPv4 de todos os servidores informáticos, assim como os endereços virtuais associados a cada instância, são apresentados na Tabela 25.

Tabela 25 - Endereçamento IPv4 dos sistemas e componentes de monitorização

Componentes e sistemas computacionais	Endereçamento IPv4	Endereço IPv4 virtual (VIP)
***cluster* de monitorização**		
Servidor zabbix 1	10.200.100.111	10.200.100.113/24
Servidor zabbix 2	10.200.100.112	
***cluster* de base de dados**		
Percona DB 1	10.200.100.127	
Percona DB 2	10.200.100.128	
Percona DB 3	10.200.100.129	
proxy cluster		
proxy 1	10.200.100.121	10.200.100.123/24
proxy 2	10.200.100.122	

Toda a arquitetura envolvente foi projetada, de forma a garantir todos os mecanismos de redundância entre os sistemas integrados nas componentes.

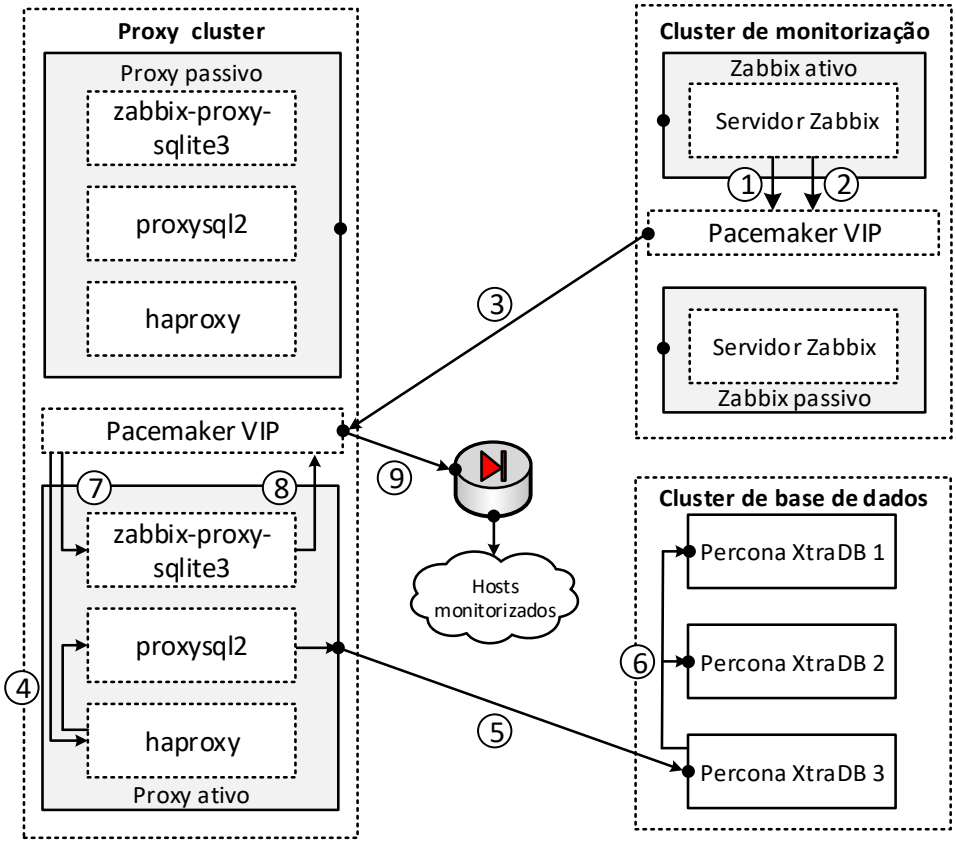


Figura 36 - Funcionamento genérico da arquitetura de monitorização implementada

A componente de monitorização recorre ao Pacemaker para responder a um endereço virtual (VIP), bem como a componente de *proxy*, que utiliza a mesma metodologia, respondendo de igual modo a um único endereço virtual.

Os servidores de armazenamento utilizam o mecanismo de sincronização denominado de Galera, de modo a criar o modelo de replicação de dados, definindo de igual modo uma arquitetura de *cluster* altamente resiliente. Todos os sistemas computacionais de armazenamento trocam a mesma informação com recurso ao Xtra Backup SST (v2), *State Snapshot Transfer*, sendo que, em casa de falha de um dos sistemas, o Galera controla o último estado desse nó e, posteriormente, sincroniza a configuração com o Xtra Backup SST (v2). Esta componente de armazenamento suporta a indisponibilidade de duas máquinas, sendo que o *proxysql* irá validar os estados de cada nó (ver secção 2.1).

O conceito do funcionamento genérico da arquitetura adotada entre componentes será apresentado tendo em conta os dois tipos de fluxos de tráfego que são originados pelo *cluster* de monitorização do Zabbix. Em seguida, com base na Figura 36, é possível detalhar os principais passos de funcionamento:

- Primeiramente, no ponto 1 é originado um *datagrama* IP com origem no endereço VIP 10.200.100.113 (Pacemaker do *cluster* de monitorização), tendo este como destino o endereço VIP 10.200.100.123 (Pacemaker do cluster do proxy). Este *datagrama* IP contém portas de origem e destino (3306), para que sejam trocados conhecimentos respetivos ao armazenamento de informação no gestor de base de dados MySQL;

- No ponto 2, é criado um *datagrama* IP com os mesmos endereços de origem e destino, descritos no ponto 1. A diferença é que este *datagrama* IP é proveniente de dados de monitorização para serem recolhidos num dado *host* e, por isso, tem como portas de origem e destino (10051) respeitantes à aplicação de zabbix-proxy-sqlite3;

- Posteriormente no ponto 3, os dois *datagramas* IP (que transportam dados para monitorização e armazenamento) são entregues no mesmo endereço IP virtual do *cluster* de *proxy*, de modo, a poderem ser encaminhados para as respetivas aplicações;

- Após a receção dos diferentes fluxos de tráfego, no ponto 4 iremos analisar como a informação respetiva aos nós de armazenamento é comutada no sistema de encaminhamento. A aplicação HAproxy está à escuta de todo o tráfego que lhe chegue com o endereço IP de destino 10.200.100.123 na porta 3306. Quando isto acontece,

redireciona o tráfego para a aplicação proxysql2 local e esta, por sua vez, encaminha o tráfego para o nó (10.200.100.129) de armazenamento que estiver eleito como primário. Neste processo o *datagrama* IP dá saída para o *cluster* de armazenamento com o endereço IP de origem (10.200.100.121) do próprio servidor ativo e não com o endereço IP virtual;

- No ponto 5 o tráfego de armazenamento é rececionado para nó de Percona escolhido como primário por parte do proxysql2. Depois dessa receção, no ponto 6 é ilustrado o mecanismo de replicação multi-*master*, sendo que o nó eleito é responsável por difundir a informação aos restantes nós constituintes do *cluster* de base de dados. Este processo é efetuado com recurso à rede e aos endereços IP de cada sistema computacional que decompõe o *cluster* de armazenamento.
- Em relação ao ponto 7, após a receção do *datagrama* IP por parte do *cluster* de *proxy*, o Pacemaker encaminha este pacote para o servidor que está ativo. Depois deste processo, com base na porta de destino, o *datagrama* IP é redirecionado para zabbix-proxy e, a informação é guardada numa base de dados auxiliar (sqlite3). Por fim, no ponto 8, o *datagama* IP é comutado com o endereço de origem 10.200.100.123 (Pacemaker VIP) e com o endereço de destino do servidor a monitorizar (ponto 9).

5.6 Arquitetura de armazenamento de dados

O estudo das tecnologias inerentes à solução para o armazenamento de dados por parte dos servidores de monitorização baseados em Zabbix, originou a conceção de um *cluster* de base de dados, que garantiu os princípios de alta disponibilidade preconizados pelo *software* Percona baseado em MySQL.

Todas as configurações serão apresentadas e divididas em duas componentes, nomeadamente na vertente da parametrização dos nós de armazenamento (Percona XtraDB *cluster*, consultar Anexo XII), e no respetivo mecanismo de encaminhamento de dados (proxySQL).

O *cluster* será composto por três sistemas computacionais, de modo a viabilizar que um dos nós de computação seja o servidor principal (*master*) para que sejam evitados possíveis problemas de *split-brain*.

O modelo relacional de base de dados implementado teve em conta alguns pressupostos que cumprem com as recomendações oficializadas pelo Galera cluster [74]. Embora o Percona

XtraDB *cluster* seja o mecanismo de gestão de base de dados (*Database Management System*, DBMS) escolhido, deve ser salientado que o mesmo recorre (nativamente) à arquitetura do Galera cluster para sincronizar a informação, de modo a que esta seja consistente entre os nós de armazenamento que decompõe o *cluster*.

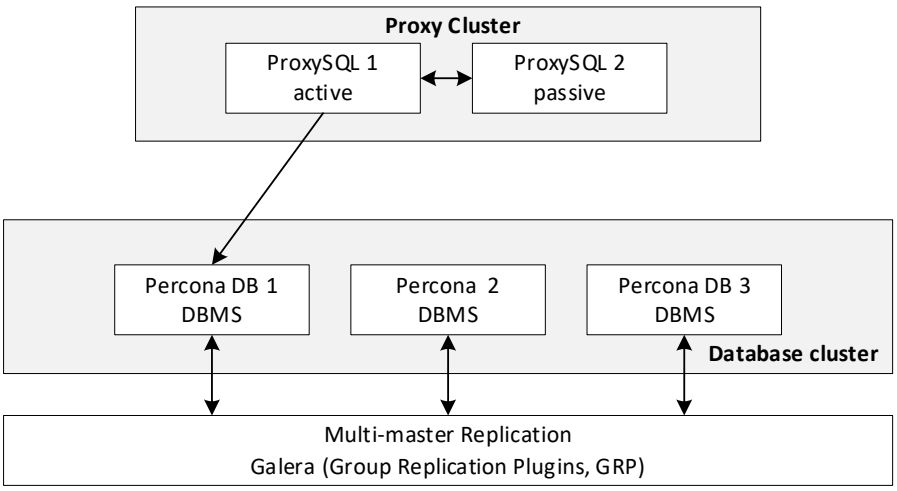


Figura 37 - Modelo de replicação de dados sugerido

Na Figura 37 é demonstrado o método de replicação de dados que, por definição, tem um funcionamento síncrono. Por outras palavras, quando a transferência de dados é confirmada, todos os nós do *cluster* passam a conter o mesmo identificador global de transação (*Global Transaction* ID, GTID). A de replicação em modo multi *master* é a mais aconselhada para *clusters* de armazenamento de alta disponibilidade [74].

Arquitetura dos nós do *cluster*

A API de replicação de dados de um nó do Galera *cluster* baseia-se em quatro elementos arquitetónicos que interagem entre si, designadamente [75]:

- O sistema de gestão de base de dados;
- A API wsrep (*write-set replication*);
- O plugin de replicação do Galera (*Galera Replication Plugin*, GRP);
- Os plugins do sistema de comunicação em grupo (*Group Communication Systems Plugins*, GCSP).

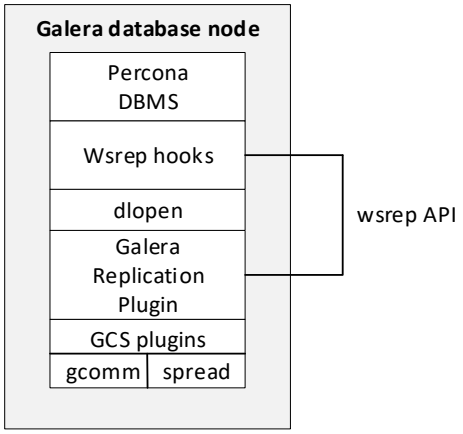


Figura 38 - Arquitetura interna de um nó de Galera *cluster*

O Galera *cluster* proporciona uma arquitetura de funcionamento bem estruturada que pode ser integrada com diferentes DBMS, conforme ilustrado na Figura 38. O seu funcionamento é bastante estruturado e conciso [75]:

- Mecanismo wsrep: analisa o estado do nó, verificando se existe alterações;
- Módulo dlopen(): é uma função que auxilia a validação dos estados do nó e da gravação dos dados pelo módulo *Wsrep hooks*;
- Plugin de replicação do Galera: verifica e certifica a replicação, de forma a alterar o estado do nó caso existam mudanças disponibilizadas pelos mecanismos anteriores;
- Camada de sistema de comunicação em grupo (GCS): trata de sincronizar os nós com a mesma configuração através de conexões TCP (porta 4567).

Funcionamento da replicação de dados do *cluster*

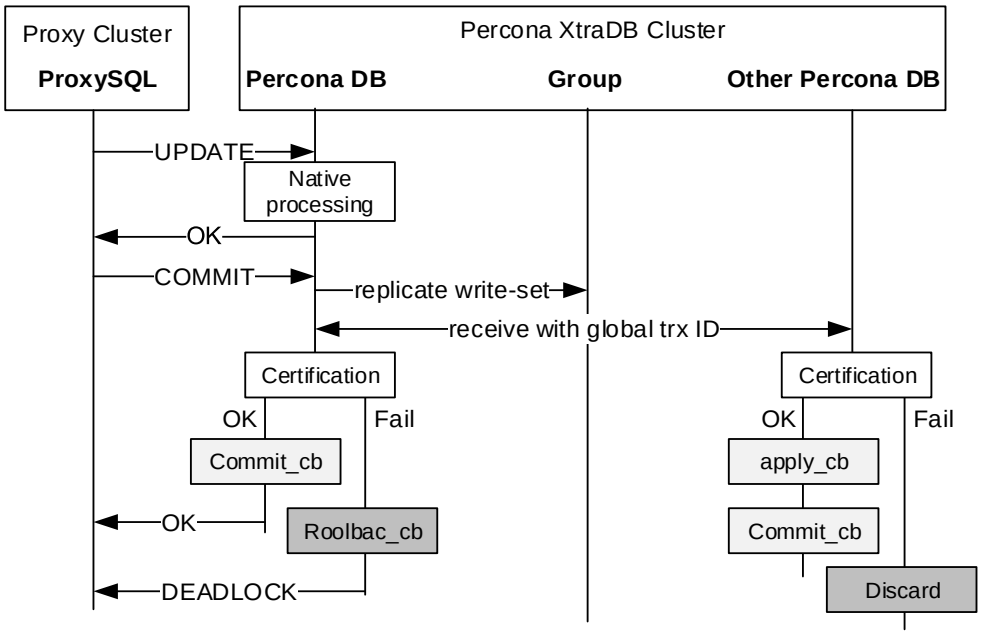


Figura 39 - Diagrama de replicação multi *master* baseada em certificação [76]

Por sua vez, o diagrama da Figura 39 descreve os mecanismos de replicação de dados entre dois nós. O mesmo esquema de funcionamento pode ser aplicado, independentemente do número de nós que decompõe um dado *cluster*.

Quando o proxySQL (cliente) emite um *commit* para o sistema de Percona DB (*Database*) primário, todas as alterações a serem realizadas são enviadas para os restantes nós associados ao *cluster* de armazenamento (*replicate write-set*).

A alteração dos dados (*write-set*) passa por uma verificação de certificação determinística. Este processo é feito localmente em cada nó associado ao *cluster*, incluindo o nó que origina a réplica de modificação da sua informação. Deste modo a componente de certificação é a entidade responsável por determinar se um nó de armazenamento pode ou não guardar os dados.

Se o teste for bem-sucedido, a transação é confirmada e o conjunto de dados será aplicado. No caso de o teste de certificação falhar, o nó eliminará os dados recebidos e todos os restantes nós do *cluster* irão utilizar o último ID de transação (*global transection* ID), garantindo assim a consistência da informação.

Modos de proteção dos dados

Por questões de retrocompatibilidade entre os recursos e o tipo de dados suportados pelo Percona XtraDB *cluster* (PXC), é empregue o mecanismo de PXC *Strict Mode*, descrevendo assim, as diferentes metodologias para validar a compatibilidade dos dados recebidos no processo de inicialização dos nós, bem como durante a execução dos mesmos. São suportadas políticas para manipular o comportamento do PXC *Strict Mode*, sendo que as mais importantes de referir são as utilizadas em modelos de replicação em grupo (*clusters*), designadamente os modos [77]:

- *enforcing*: caso uma validação falhe durante o processo de inicialização de um nó de armazenamento, este não arranca o processo *mysqld* e gera um erro no sistema de *logs*. Se uma validação falhar durante a execução do sistema, a operação é abortada despoletando um erro.
- *master*: o funcionamento é semelhante ao modo *enforcing* com a diferença que o procedimento da validação após uma falha, não bloqueia as tabelas da base de dados, permitindo na mesma a escrita destes. Este modo é recomendado em *clusters* onde as operações de gravação (*write-set*) são isoladas num único nó de armazenamento (modelo *master-slave*).

Recuperação de dados em caso de falha

O PXC permite utilizar funcionalidades SST (*State Snapshot Transfer*) para recuperar a informação armazenada em caso de falha de um ou mais nós que constituem um *cluster* (Anexo XI).

As principais metodologias SST são as seguintes:

- mysqldump: este método lógico é o mais lento na transferência de dados e não garante a disponibilidade do nó que transfere os dados (*read-lock*). No entanto, é o único mecanismo que pode ser utilizado sem inicializar o nó que perde o sincronismo das réplicas;
- rsync: é um mecanismo de transferência de dados ao nível físico com a mesma limitação do mysqldump (re*ad lock)*. Consegue ser de todos, o mecanismo mais rápido a sincronizar os dados;

- xtrabackup: foi o modelo adotado, porque dos mecanismos apresentados anteriormente, este é o único que permite a disponibilidade do nó de transferência para possíveis leituras (*read-only*). Cabe salientar que este mecanismo ocorre ao nível físico, como o rsync, e por isso obriga à iniciação do serviço myqld do nó que está com problemas.

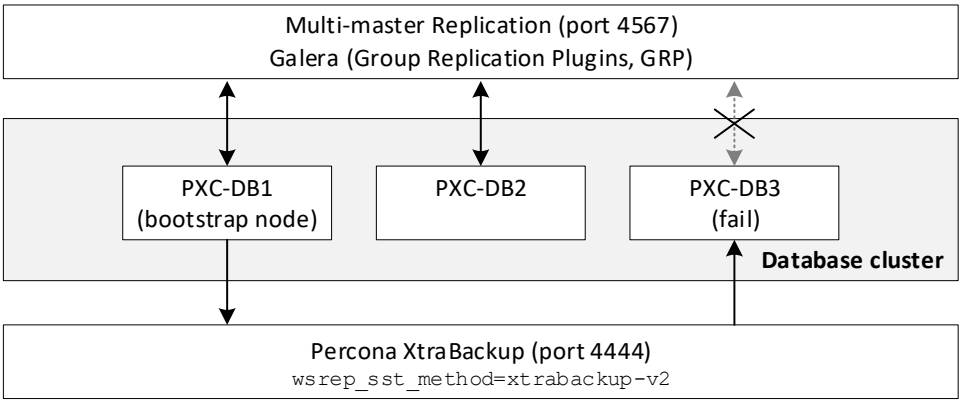


Figura 40 - Recuperação de falhas com a funcionalidade XtraBackup

Na Figura 40 é demonstrado um exemplo do *cluster* que irá ser implementado. A título de exemplo, podemos verificar uma falha ocorrida no terceiro nó de base de dados (PXC-DB3).

A replicação em modo multi-*master* (GRP) deteta uma inconsistência, sendo que o terceiro nó (recetor) deixa de fazer parte do *cluster*. Isto origina que o serviço *mysqld* pare no nó onde é detetada a indisponibilidade. Com a inicialização manual do processo *mysqld* no sistema recetor, o primeiro nó (emissor) que originou a sincronização do *cluster* (*bootstrap node*), recorre ao mecanismo Xtrabackup (v2.4), para transferir a sua configuração. Concretamente, este procedimento é composto por duas etapas em simultâneo:

- Na primeira etapa com base na sequência de *log* (*Log Sequence Number*, LSN) é iniciado um processo de cópia e transmissão de arquivos, podendo este levar algum tempo, visto que os dados podem estar a ser alterados nesse momento, dependendo também da análise da segunda etapa auxiliar;
- Paralelamente são avaliados os arquivos de log transacionais (*InnoDB*) que estão efetivamente guardados com base no algoritmo *round-robin*. Nesta segunda fase são determinadas as configurações mais recentes, sendo dado a conhecer à primeira etapa as últimas transações recebidas pelo nó transmissor.

5.6.1 Encaminhamento do fluxo de dados de armazenamento

O conceito de encaminhamento dos dados entre a componente de monitorização e a componente de armazenamento foi dissociado, com o objetivo de tirar maior partido computacional e de resiliência na solução exposta (capítulo 5). Pelo facto de esta conceção obrigar a modelos de *proxys*, foram estudados os dois *softwares*, que em consonância formam um modelo de *proxy cluster*.

Inicialmente, a ferramenta HAproxy (ver Anexo XIV), foi utilizada para o gestor de base de dados MariaDB. Era uma das possibilidades mais fiável para encaminhar dados para os nós de armazenamento, em arquiteturas *open-source*. Contudo, e com as limitações descobertas na implementação do MariaDB (consultar secção 5.3), elegeu-se o modelo baseado em Percona XtraDB *cluster*, o qual, define o procedimento de encaminhamento com auxílio do *software* proxySQL (ver Anexo XV).

Embora o proxySQL permita fazer o encaminhamento dos dados recebidos pelo Zabbix de forma "direta", manteve-se o HAproxy para reencaminhar esses dados entre o Zabbix e o proxySQL. Na verdade, a grande vantagem prende-se pela visualização *web*, onde se verifica rapidamente os servidores que estão ativos e a informação do número de ligações MySQL, entre outras métricas.

Ainda assim, por uma questão de simplicidade de configuração, o modelo final desta implementação, poderá levar a descontinuar a utilização do HAproxy, sendo que seria menos uma aplicação a gerir, bem como, diminuiria as latências e possíveis pontos críticos de falha.

Funcionamento genérico da arquitetura do *proxy cluster*:

A perceção sobre o funcionamento dos mecanismos de encaminhamento entre a componente de monitorização e a componente de armazenamento recai sobre quatro passos fundamentais (Figura 41):

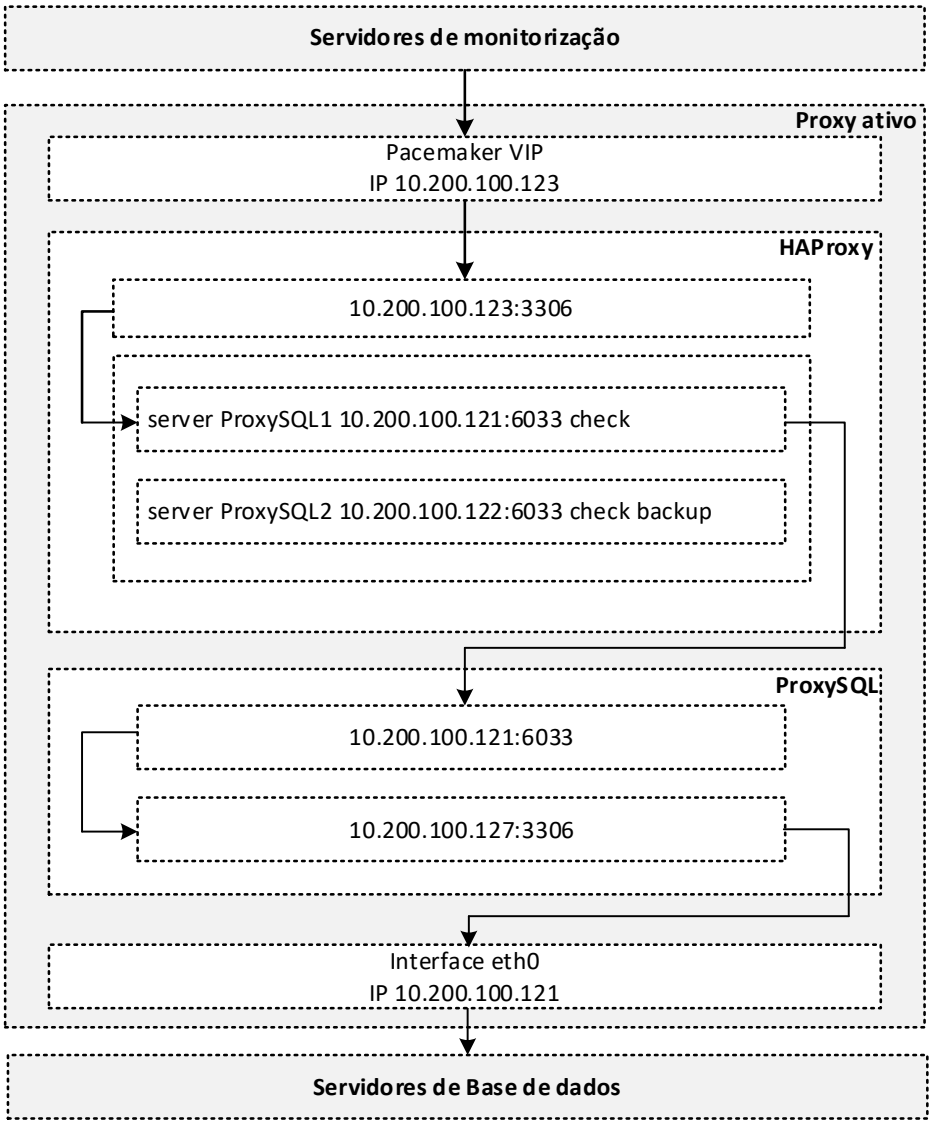


Figura 41 - Processo de encaminhamento do fluxo de tráfego MySQL

- Inicialmente o fluxo de tráfego MySQL é transmitido pelos servidores de monitorização, com base no endereço de destino do Pacemaker. O *proxy* que estiver ativo, irá responder ao respetivo endereço VIP (10.200.100.123).
- A aplicação HAproxy recebe o tráfego com destino ao endereço IP 10.200.100.123 na porta 3306 e redireciona o mesmo, para a lista de servidores instanciados (proxySQL), também descrita no ficheiro de configuração (`haproxy.cfg`). Apenas um servidor proxySQL que estiver configurado como ativo (*check*), irá receber o fluxo de tráfego MySQL, sendo que neste passo o endereço de destino/porta é alterado pela primeira vez. Caso exista uma falha de comunicação com o servidor proxySQL ativo, o tráfego seria encaminhado para o servidor proxySQL redundante (*check backup*).

- Quando o tráfego é analisado pelo proxySQL, este faz também uma operação de redireccionamento semelhante ao HAproxy. Deste modo, todo o tráfego que tiver como destino o endereço IP 10.200.100.121 na porta 6033 (porta nativa de redireccionamento de tráfego MySQL do *software* PorxySQL), será trocado pela segunda vez o endereço IP de destino/porta, de forma a alcançar o nó de armazenamento atribuído (consultar Figura 42).
- O pacote IP dá saída do servidor de *proxy* ativo com o endereço IP de origem 10.200.100.121 e com o endereço IP de destino 10.200.100.127.

Consulta e alteração de dados:

A definição de escritas e leituras, por parte do proxysql, é realizada por omissão com identificadores (*host group* ID). Cada identificador agrega diferentes comportamentos como ilustra a Figura 42.

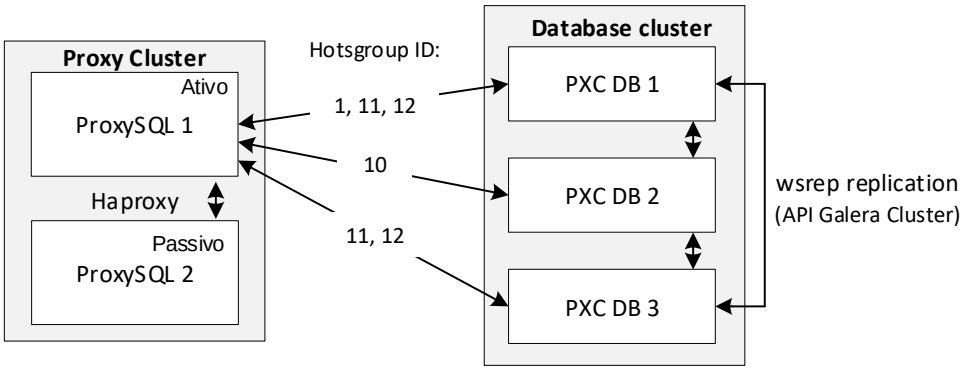


Figura 42 - Processo de encaminhamento de dados via proxySQL2

O nó de ativo é considerado aquele que possibilita a replicação dos dados (ID 1). Por sua vez, apenas um nó poderá realizar o processo de escrita (ID 10). Apenas dois nós sãoi eleitos para efeitos de consulta dos dados (ID 11). O mecanismo de *backup writer* (ID 12), garante a resiliência dos nós de escrita, caso o nó ativo atinja o número máximo de ligações, ou por indisponibilidade do sistema. Em caso de falha de um ou mais nós de armazenamento, existe um identificador (ID 13) reservado (consultar Tabela 26).

Tabela 26 - Identificadores de consulta e alteração de dados

Host group	*Host group ID*	*Host name*	*Port*
active	1	PXC DB 2	3306
writer	10	PXC DB 2	
reader	11	PXC DB 1 PXC DB 3	
Backup writer	12	PXC DB 1 PXC DB 3	
offline	13		

5.6.2 Mecanismos de resiliência

A solução adotada foi desenhada para cumprir com os mecanismos de alta disponibilidade. Nativamente, o Percona XtraDB *cluster* utiliza a tecnologia protocolar do Galera cluster para proteger a consistência dos dados entre os nós de armazenamento, bem como garantir a recuperação da informação em caso de falha, com o recurso às metodologias SST (ver secção 5.6.1).

No que concerne ao modo de encaminhamento dos dados, o *cluster* de *proxys* foi concebido de modo a acautelar um servidor redundante, protegendo esta componente em caso de falha. Para isso, foram estudadas e implementadas duas ferramentas, nomeadamente, o Keepalive e o Pacemacker.

***Cluster* de base de dados**

Depois da configuração do PXC devem ser executados testes de resiliência, a fim de entender se o mesmo está a ter o comportamento estudado anteriormente (secção 5.6.1).

```
mysql> show status like 'wsrep%';
+----------------------------+--------------------------------------+
| Variable_name              | Value                                |
+----------------------------+--------------------------------------+
| wsrep_local_state_uuid     | c2884438-834d-11e2-0800-03c9c68e44de |
| wsrep_local_state          | 4                                    |
| wsrep_local_state_comment  | Synced                               |
| wsrep_cluster_size         | 3                                    |
| wsrep_cluster_status       | Primary                              |
| wsrep_connected            | ON                                   |
| wsrep_ready                | ON                                   |
+----------------------------+--------------------------------------+
```

Figura 43 - Validação do estado do número de nós do PXC

Primeiramente, deve ser analisado dentro da base de dados MySQL, se um dos nós de armazenamento, consegue identificar os membros associados ao *cluster* (ver Figura 43).

Posteriormente, é aconselhado verificar se as réplicas (GRP) estão a ser trocadas devidamente. Para isso, basta criar uma tabela em MySQL, de modo a testar se a mesma é copiada para os restantes nós de armazenamento do *cluster*. Para tal, procedeu-se aos seguintes passos:

1. Criação de uma base de dados de testes chamada "percona" no segundo nó do PXC:

```
mysql@perconadb2> CREATE DATABASE percona;
Query OK, 1 row affected (0.01 sec)
```

2. Criação de uma tabela "exemplo" no terceiro nó do PXC:

```
mysql@perconadb3> USE percona;
Database changed

mysql@perconadb3> CREATE TABLE exemplo (node_id INT PRIMARY KEY, node_name VARCHAR(30));
Query OK, 0 rows affected (0.05 sec
```

3. Inserção de valores na tabela exemplo no primeiro nó do PXC:

```
mysql@perconadb1> INSERT INTO percona.exemplo VALUES (1, 'percona1');
Query OK, 1 row affected (0.02 sec)
```

4. Comprovação de todas as alterações realizadas no segundo nó do PXC:

```
mysql@perconadb2> SELECT * FROM percona.exemplo;
+---------+-----------+
| node_id | node_name |
+---------+-----------+
|       1 | percona1  |
+---------+-----------+
1 row in set (0.00 sec)
```

Recuperação de dados em caso de falha

Numa primeira abordagem, qualquer nó que apresente problemas de sincronização poderá ser "forçado" a inicialização do serviço `mysql`. Por norma, é suficiente em casos de problemas de rede. Caso o nó não consiga sincronizar com o *cluster* pode ser realizado um `mysqldump`, de forma manual com a configuração de um dos nós ativos do *cluster*.

```
[root@perconadb1~]# vi/var/lib/mysql/grastate.dat

# GALERA saved state
version: 2.1
uuid:    a70ad193-1da6-11ea-976d-d392f87771ff
seqno:   -1
safe_to_bootstrap: 1
```

Figura 44 - Arrancar um nó pelo *bootstrap* manualmente

Em casos mais extremos em que existe um erro replicado para todos os nós, o recomendado é parar todos os serviços `mysql`. Depois desse procedimento, deve-se compreender qual foi o último nó a ter problemas. Identificado o nó, deve-se então "forçar" manualmente a sua inicialização através do ficheiro de *bootstrap*. Com esta metodologia, o nó irá conseguir proceder à última transação efetiva, sendo que na maior parte das vezes o erro é corrigido. Inerentemente, o campo `safe_to_bootstrap` vem com o valor "0" e deve ser alterado para o valor "1" conforme a configuração representada na Figura 44

Habitualmente, este tipo de erros de replicação é proveniente do servidor (cliente) que gera dados incompatíveis para armazenamento, sendo estes detetados pelo mecanismo de retrocompatibilidade (*strict-mode enforcing*). É recomendado analisar sempre as duas entidades da arquitetura, concretamente o servidor de monitorização Zabbix e os nós de armazenamento do PXC.

***Cluster* de encaminhamento**

Os proxys contêm um papel preponderante em todo o funcionamento da solução (ver topologia na Figura 36). Mediante esse facto, no primeiro cenário de estudo foi implementada a ferramenta Keepalived (ver Anexo XVII) para conseguir garantir a alta disponibilidade (*High-availability*) através de dois sistemas computacionais. Após alguns testes da aplicação em questão, compreendeu-se que era necessário um controlo ao nível de funcionamento do serviço (processos em execução), e que não fosse analisada a disponibilidade dos sistemas baseada apenas na rede (camada 3 do modelo TCP/IP).

Através das limitações anteriormente citadas, no segundo caso de estudo, foi configurada a aplicação Pacemaker, a qual, colmatou a restrição do Keepalived. Nesta secção serão retratadas as configurações realizadas nas duas ferramentas implementadas, sendo dado o maior relevo à aplicação eleita (Pacemacker) visto ter ficado em produção (ver Anexo XVIII).

Funcionamento genérico do Keepalived

Dada a necessidade de garantir o princípio da resiliência dos dois sistemas informáticos que integram a aplicação HAproxy, procedeu-se à inclusão do *software* Keepalived numa fase inicial.

Tabela 27 - Especificações para configuração do Keepalive (cenário 1)

Endereço IPv4	Nome do *host*	Aplicações
10.200.100.121	*proxy1*	HAproxy
10.200.100.122	*proxy2*	HAproxy
10.200.100.123	VIP proxy-cluster	Keepalived

O princípio de funcionamento do Keepalived é estruturado no módulo *kernel* do servidor Linux virtual (*Linux Virtual Server*, LVS), concedendo a hipótese de recorrer a técnicas de servidor IP virtual (IP *Virtual Server*, IPVS).

Relacionado ao conceito mencionado anteriormente, o Keepalived é fundamentado pelo protocolo VRRP (*Virtual Router Redundancy Protocol*) para eleger um dos sistemas computacionais, baseando-se em dois estados, particularmente o estado *master* ou o estado *backup*. Para que dois ou mais sistemas autónomos possam trocar dados, de modo a verificar a disponibilidade destes, é utilizado um mecanismo de tolerância a falhas (*failover*), o qual, recorre à inclusão de um endereço IP virtual (VIP).

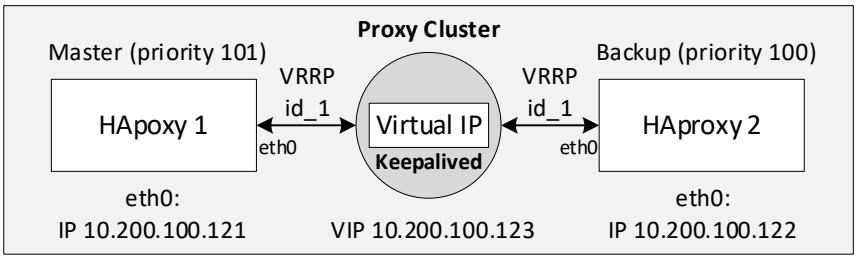


Figura 45 - Funcionamento genérico do Keepalived

Conforme ilustrado na Figura 45, a negociação do processo de eleição entre dois endereços associados às interfaces físicas de cada servidor *proxy*, é efetuada com recurso a um terceiro endereço IP, nomenclado de endereço VIP (`virtual_ipaddress`), o qual é definido no ficheiro de configurações do Keepalived (`keepalived.conf`).

Ainda no ficheiro de configurações do Keepalived é determinado o critério de prioridade (`priority`) para eleger a máquina que responde ao endereço VIP. O sistema de *proxy* que tiver atribuído o valor numérico mais alto neste parâmetro é considerado ativo (*master*). Por sua vez o servidor que tiver configurado o valor numérico mais baixo passa a estado inativo (*backup*).

O domínio ao qual pertencem os dois servidores *proxy* é reconhecido pelo identificador virtual do *router* (`virtual_router_id`), devendo este ser comum para que ambos comuniquem através do mesmo endereço virtual.

O Keepalived permite ainda ter várias instâncias VRRP (`vrrp_instance`), possibilitando que várias máquinas possam responder a endereços VIP distintos, ou a mais que um endereço.

Funcionamento genérico do Pacemaker

Embora a arquitetura aplicacional do Pacemaker seja totalmente diferente do Keepalived, o conceito de *failover* é igualmente preconizado com recurso a um endereço VIP.

Tabela 28 - Especificações para configuração do Pacemaker (cenário 2)

Endereço IPv4	Nome do *host*	Aplicações
10.200.100.121	*proxy1*	HAproxy
10.200.100.122	*proxy2*	HAproxy
10.200.100.123	VIP proxy-cluster	Pacemakwe

Ao nível arquitetónico, o Pacemaker propõe uma estrutura com bastantes conceitos teóricos, embora ao nível de configurações seja imensamente mais simples. O seu funcionamento tem por base controlar os processos aplicacionais que pretendemos agregar a um *cluster* de servidores. Durante um evento de falha (*failover*) no servidor *proxy* ativo, o processo deverá ser interrompido e ser iniciado no servidor *proxy* redundante.

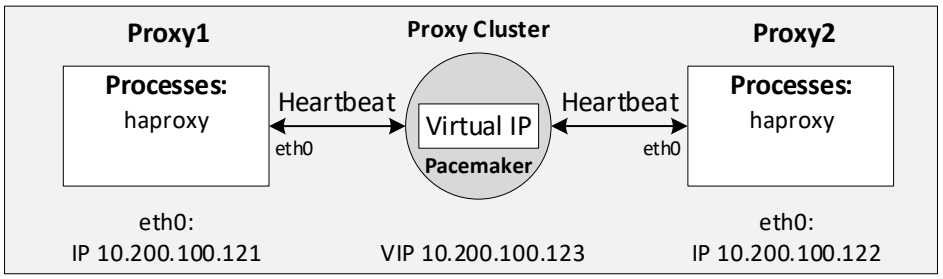


Figura 46 - Funcionamento genérico do Pacemaker

A forma como o Pacemaker controla os processos, provém da infraestrutura de baixo nível da aplicação que pode recorrer a diferentes sistemas de comunicação em grupo [78]:

- Corosync (*daemon* `pcs`);
- CMAN (*cluster Manager*);
- Heartbeat.

O Heartbeat é o mecanismo empregue pelo Pacemaker para despoletar as informações do *cluster*, nomeadamente, as mensagens de controlo dos processos (`haproxy`) e as mensagens de associação dos membros pertencentes ao *quorum* [78](Figura 46).

5.7 Arquitetura de monitorização

A arquitetura proposta para a consulta de dados é baseada no sistema de monitorização Zabbix, consolidando a interoperabilidade entre sete servidores (inclusive), conforme ilustrado na Figura 35. Do ponto de vista dos servidores Zabbix, é indispensável o entendimento dos modelos e componentes que sustentam esta solução.

Conforme foi anteriormente referido, o Zabbix recorre a um modelo de base de dados, onde são armazenadas as informações respetivas à monitorização dos sistemas informáticos. Para encaminhamento desta informação, foram previamente configurados dois servidores *proxy*.

No entanto, para recolher os dados de monitorização na rede, recorreu-se ao modelo de Zabbix-*proxy*, o qual oferece a mais-valia do balanceamento de carga computacional, bem como a otimização do desempenho dos servidores principais que integram o Zabbix, garantindo ainda, a escalabilidade da solução no futuro.

Os servidores Zabbix irão apenas suportar a aplicação. Os restantes componentes foram anteriormente segregados e parametrizados no *cluster* de base de dados (secção 5.6), bem como no *cluster* de *proxys*.

Os *proxys* Zabbix podem ainda ser distribuídos por diferentes redes em sistemas informáticos dedicados com baixos recursos computacionais, consoante a dimensão da rede e do aumento dos seus sistemas integrantes.

Todo o processo de instalação e configuração dos servidores Zabbix e dos servidores Zabbix-*proxys* será apresentado no Anexo XIX e Anexo XXI.

5.7.1 Encaminhamento do fluxo de dados de monitorização

O encaminhamento de informação de monitorização por parte do servidor Zabbix pode ser feito com recurso a mecanismos de *proxy*. Tendo em conta que a rede da U. Porto integra em média quinhentos e noventa e cinco sistemas informáticos e mil setecentos e noventa e três serviços, torna-se relevante utilizar uma solução de modelo computacional distribuído, que consiga ser complementada de forma a escalar mediante o crescimento da rede.

O mecanismo do Zabbix-*proxy* alia a possibilidade de distribuir a carga computacional dos servidores Zabbix, garantindo a hipótese de criar um ou mais servidores de *proxy* em diferentes redes, proporcionando também uma resposta mais eficaz no que respeita à latência da rede de dados.

Dentro desse paradigma, existem dois possíveis cenários, os quais são preconizados pela comunidade Zabbix SIA. Sob o primeiro ponto de vista, a base de dados do Zabbix-*proxy* pode ser remota ou local ao próprio servidor.

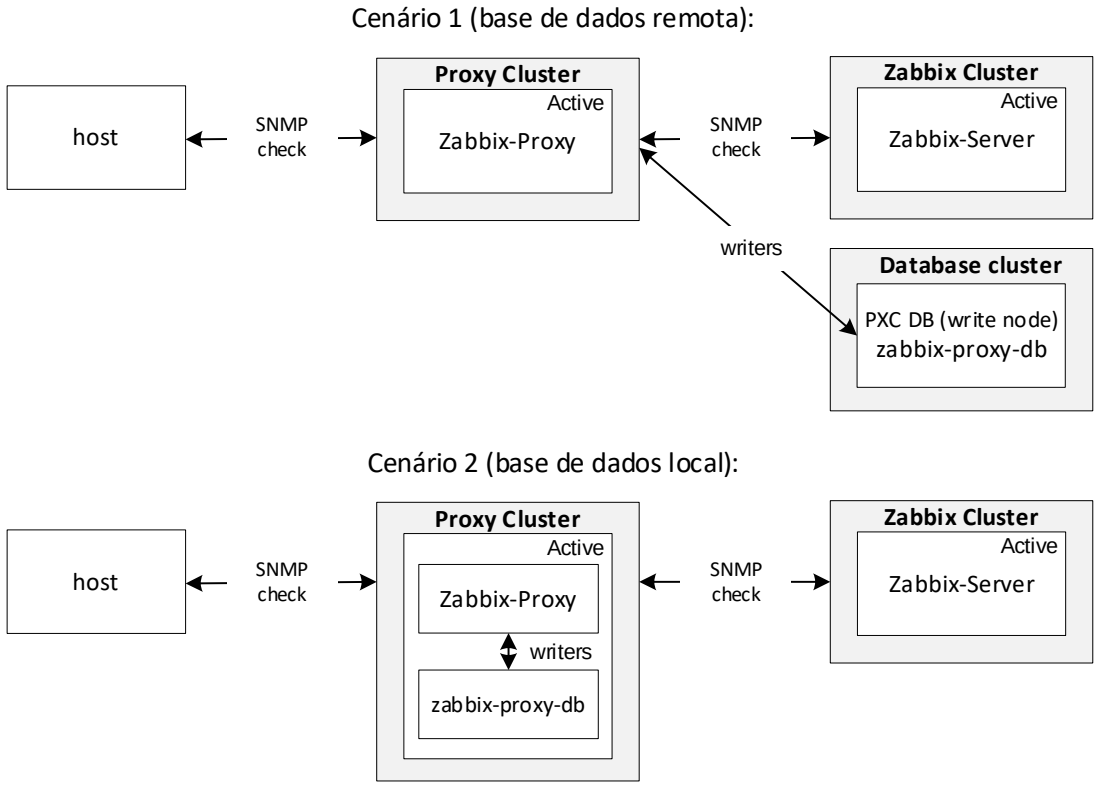


Figura 47 - Topologia da base de dados remota *versus* local

Ambos os cenários apresentados na Figura 47 foram testados durante o desenvolvimento da solução, concluindo que a utilização de uma base de dados remota para o Zabbix-*proxy*, não é efetiva. Foi criada uma base de dados com recurso ao *script* de `zabbix-proxy-mysql`, constatando-se a enorme latência na adição de um *host* e do serviço SNMP associado, bem como, os *checks* aos serviços monitorizados eram extremamente morosos a serem despoletados, isto porque a performance do Zabbix-*proxy* era diminuta.

O segundo cenário passou por solucionar a carência da latência no processo de consultas à base dados do Zabbix-*proxy*, bem como a melhoria significativa do *required performance* (valores analisados por segundo, VPS). A criação de uma base de dados local do tipo SQLite solucionou eficazmente a demora do procedimento citado, resolvendo ainda a problemática de poder existir mais que um Zabbix-*proxy*.

Compreendeu-se que a partilha de uma base de dados comum pode originar inconsistências, pois um dado Zabbix-*proxy* deixa de ter conhecimento das alterações realizadas por outro sistema computacional com a mesma função.

A proposta e a solução para as questões referenciadas anteriormente, passa por reutilizar os *proxys* de encaminhamento (ver Tabela 29) do *cluster* de armazenamento (PXC), instalando e configurando a aplicação Zabbix-*proxy* com recurso ao modelo de base de dados SQLite (ver Anexo XXI).

Tabela 29 - Servidores utilizados para a configuração do Zabbix-*proxy*

Endereço IPv4	Nome do *host*	Aplicações	Componente
10.200.100.121	*proxy1*	zabbix-proxy-sqlite	*proxy cluster*
10.200.100.122	*proxy2*	zabbix-proxy-sqlite	

O modelo distribuído de Zabbix-*proxy* pode ainda ser parametrizado em diferentes modos de operação conforme ilustra a Figura 48.

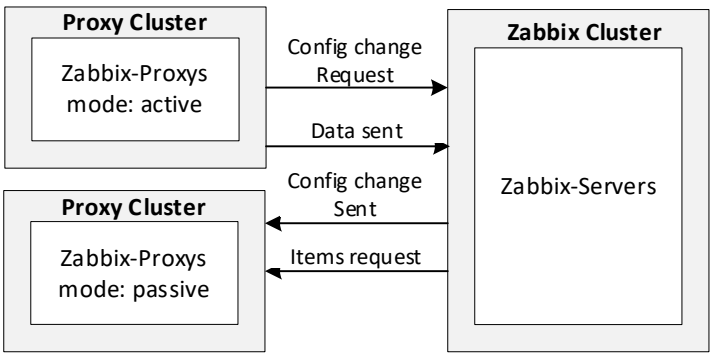


Figura 48 - Fluxo de dados em modo ativo e passivo no Zabbix-*proxy*

O modo ativo permite que o Zabbix-*proxy* seja totalmente independente do servidor Zabbix, isto é, a recolha dos dados de monitorização é realizada pela Zabbix-*proxy*. O grande proveito deste método é poder distribuir a carga computacional, pois o servidor Zabbix apenas recebe a alteração das configurações (*Data Sent*) que é enviada pelo Zabbix-*proxy*.

Por sua vez, o modo passivo aufere um comportamento que não melhora o desempenho do servidor Zabbix. Concretamente o servidor Zabbix envia todas as alterações de configurações ao Zabbix-*proxy* (*Configuration change sent*), sendo que neste tipo de encaminhamento o Zabbix-*proxy* requer menos especificações ao nível de *hardware*. A grande desvantagem recai na escalabilidade, visto que quantos mais itens (*items request*) forem adicionados a um ou mais *hosts*, maior é a latência entre a origem do sistema monitorizado e o servidor Zabbix.

A solução em redes de grande dimensão, como é o caso da rede da U. Porto, passou por configurar o modo ativo do Zabbix-*proxy* nos dois sistemas do *cluster*, de forma a conseguir monitorizar um *host* sem recorrer a pedidos de monitorização do servidor Zabbix principal.

O modo ativo permite parametrizar diferentes variáveis que determinam o tempo que a informação é guardada na base de dados local em caso de falha do *cluster* de servidores Zabbix, bem como, a periodicidade com que a alteração dos dados é enviada, entre muitos outros. Estes e outros critérios associados à performance do Zabbix-*proxy* são apresentados na secção 5.7.3.

5.7.2 Modelo de resiliência

O modelo de resiliência adotado para os sistemas computacionais que alojam o Zabbix e Zabbix-*proxy*, segue o mesmo princípio teórico da aplicação Pacemaker, o qual foi retratado na configuração do *cluster* de *proxys* (ver secção 5.6.2).

Servidores Zabbix

Para formar a componente de *cluster* Zabbix (Figura 49), foi integrada a ferramenta Pacemaker, a qual permitiu adicionar os dois serviços aplicacionais instalados em cada servidor Zabbix (ver Anexo XX).

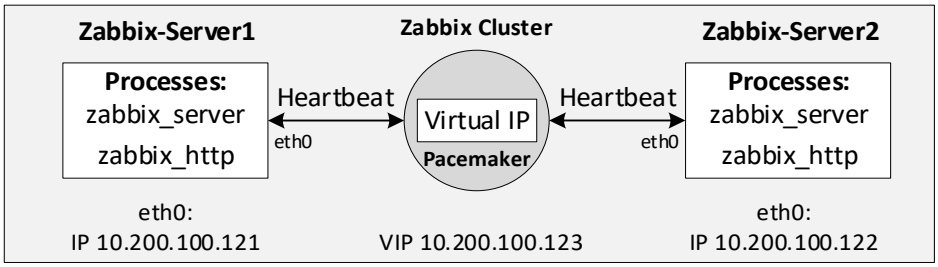


Figura 49 - Funcionamento do Pacemaker no *cluster* Zabbix

Um ponto que é importante a salientar é a periodicidade dos processos agregados ao Corosync, deste modo, deve ser considerado que o primeiro serviço a ser desativado é o `zabbix_server`, sendo que, no procedimento inverso, este deve ser o primeiro serviço a ser ativado quando o servidor inicia ou é realizado o *restart* ao serviço em questão. Este serviço cria bastantes ligações à base de dados e, por isso, pode criar problemas de *split brain* nos sistemas caso esta recomendação não se verifique.

Servidores Zabbix-*proxy*

À semelhança das configurações realizadas para a parametrização do Pacemaker na secção 5.6.2, foi apenas agregado o processo `zabbix_proxy` aos dois sistemas computacionais que decompõem o *cluster* de *proxys* (ver Anexo XXI - secção 2).

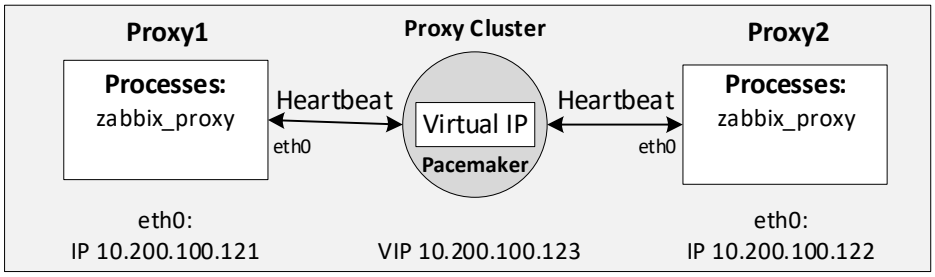


Figura 50 - Adição de serviços ao Pacemaker do *cluster* de *proxys*

Neste tipo de topologia implementada, fica percetível o entendimento teórico dos modos de funcionamento escolhidos, tendo como auxílio as ilustrações (Figura 49 e Figura 50).

O modo de operação dos Zabbix-*proxy*, neste caso, só poderia ser definido com ativo. Desta forma os dois servidores Zabbix-*proxy* vão validar primeiramente as configurações ao *cluster* Zabbix. Na falha de um dos sistemas Zabbix-*proxy*, a sincronização é sempre bem executada.

Por outro lado, em modo passivo, o servidor Zabbix solicita as configurações ao Zabbix-*proxy*, e este pode estar com as configurações diferentes gerando inconsistências na base de dados.

Ao nível do modelo de resiliência o servidor proxy1 é preferencialmente o que irá responder ao endereço VIP caso se verifique o normal funcionamento deste sistema.

5.7.3 Otimização e performance

Para otimizar um sistema computacional existem inúmeras questões que devem ser analisadas. No entanto, o sistema de monitorização Zabbix tem especificações globais que podem ser ajustadas à medida que o número de equipamentos e respetivos serviços que pretendemos monitorizar aumenta.

Tendo em conta o estudo realizado anteriormente para os requisitos funcionais dos servidores (capítulo 5.4), bem como do modelo de armazenamento de alta disponibilidade (capítulo 5.6), torna-se fundamental melhorar o desempenho da componente de monitorização.

Depois de efetuados testes aos servidores Zabbix, compreendeu-se que os valores por omissão definidos no ficheiro de configuração (`zabbix_server.conf`), não são suficientes para suportar o normal funcionamento do sistema.

Como boas práticas gerais, enumera-se os seguintes procedimentos de modo a minimizar os requisitos computacionais por parte dos sistemas integrantes da solução proposta:

- Monitorização dos parâmetros que são apenas necessários;
- Otimização do intervalo entre verificações de todos os itens;
- Alteração dos parâmetros padrão, nomeadamente em *templates*, *triggers*, entre outros;
- Melhorar os parâmetros de limpeza da informação da base de dados;
- Não monitorizar parâmetros que retornam a mesma informação.

À data da elaboração deste relatório, o sistema de monitorização está a suportar vinte e nove mil trezentos e quarenta e seis *items*, oitenta e um *hosts*, cento e cinquenta e um *templates* e onze mil trezentos e cinco *triggers*. Os valores originais foram parametrizados para suportar pelo menos mil *hosts* e cem mil itens. Esses ajustes foram realizados tanto nos servidores Zabbix, como nos servidores Zabbix-*proxy*, os quais serão apresentados nas secções seguintes.

Servidores Zabbix

Tabela 30 - Parâmetros para melhoria de desempenho do Zabbix

Parâmetros	Intervalo de configuração	Descrição
Memoria cache		
`CacheSize`	128 *Kilobytes* - 8 *Gigabytes*	Define o valor global da memória cache utilizada pela aplicação Zabbix
`CacheUpdateFrequency`	1-3600 segundos	Valor que especifica o intervalo de tempo de atualização da memória cache
`HistoryCacheSize`	128 *Kilobytes* - 2 *Gigabytes*	Descreve o tamanho da memória cache respetiva ao armazenamento do histórico de dados
`HistoryIndexCacheSize`	128 *Kilobytes* - 2 *Gigabytes*	Define o valor da memória cache respetiva à indexação de dados do histórico
`TrendCacheSize`	128 *Kilobytes* - 2 *Gigabytes*	Valor que define a memória compartilhada para informação de armazenamento baseada em tendências.
`ValueCacheSize`	0.128 *Kilobytes* - 64 *Gigabytes*	Define o valor total de memória cache relativa a dados do histórico de um dado item
Processos internos		
`StartPollers`	0 – 1000 processos	Referem-se ao número de processos internos de pesquisa dos diferentes tipos e modos de monitorização do sistema
`StartPollersUnreachable`	0 – 1000 processos	Número de processos responsáveis pelos sistemas monitorizados que se encontram indisponíveis
`StartTrappers`	0 – 1000 processos	Processos para monitorização ativa com base em *Trappers*
`StartPingers`	0 – 1000 processos	Valor de ajuste para processos de descoberta de *hosts* via ICMP
`StartHTTPPollers`	0 – 1000 processos	Número de processos para monitorização *Web*
`StartDBSyncer`	1 – 100 processos	Define o número de processos que sincronizam a base de dados do Zabbix

O ajuste de desempenho dos servidores Zabbix foi efetuado com base em dois grupos genéricos de parâmetros (Tabela 30):

- A memória cache aplicacional deve ser personalizada para melhorar o tempo de atualização dos dados do histórico (`CacheSize`, entre outros);
- Os processos internos necessitam de ser aumentados consoante a percentagem de utilização face ao incremento dos serviços de monitorização.

Os parâmetros apresentados na Tabela 30 foram reajustados e podem ser consultados os respetivos valores no Anexo XXII. Estes ajustes foram calculados com base na própria monitorização estatística dos servidores Zabbix e dos seus agentes, com auxílio de *Templates* genéricos (*Templates*: App Zabbix Server, OS Linux by Zabbix *agent*), os quais são disponibilizados pela interface gráfica da aplicação.

A monitorização estatística foi concretizada em diferentes espaços temporários, no entanto, aconselha-se a monitorizar os processos internos do Zabbix durante um mês, fazendo leves ajustes periódicos, de modo a não desperdiçar recursos de *hardware* dos sistemas computacionais.

Servidores Zabbix-*proxy*

O modo de funcionamento adotado para o Zabbix-*proxy* irá determinar a performance que o mesmo poderá ter. Sobre este ponto de vista, podemos constatar que o funcionamento em modo passivo irá sofrer ajustes de desempenho no próprio servidor Zabbix, baseado nos parâmetros apresentados na Tabela 30.

Por outro lado, o processo de comunicação entre o Zabbix-*proxy* e o servidor Zabbix pode ser preponderante no que diz respeito à latência que o modo passivo poderá provocar. Nesse contexto os parâmetros apresentados na Tabela 31 devem ser ajustados mediante a dimensão e os requisitos do ambiente de monitorização, destacando-se que as configurações ao nível passivo, são sempre realizadas no servidor Zabbix.

Concretamente, na solução em questão, o modo de funcionamento do Zabbix-*proxy* está implementado num modelo de funcionamento ativo. A melhoria do desempenho, conforme abordado anteriormente, é bastante significativa, no entanto carece de ajustes no próprio servidor Zabbix-*proxy*. A parametrização do desempenho no Zabbix-*proxy* divide-se em três grupos fundamentais:

- A otimização do processo de comunicação entre o Zabbix-*proxy* e o servidor Zabbix;
- Definir o modo como os dados provenientes da monitorização são armazenados, bem como a periodicidade com que os mesmos são enviados ao servidor Zabbix;
- O ajuste de parâmetros de desempenho específicos (`zabbix_proxy.conf`), sendo que estes parâmetros são iguais aos do servidor Zabbix (Tabela 30).

Tabela 31 - Opções para melhoria de comunicação do Zabbix-*proxy*

Parâmetros	Intervalo de configuração	Descrição
Zabbix- *proxy* em modo ativo		
`proxyLocalBuffer`	1 - 720 horas	Este parâmetro permite definir um valor em horas que especifica o tempo que o *proxy* irá guardar os dados, independentemente de serem enviados ou não ao servidor Zabbix.
`proxyOfflineBuffer`	1 - 720 horas	Definição do tempo em que o proxy guarda a informação na base de dados local, apenas em caso de falha no processo de envio ao servidor Zabbix
`HeartbeatFrequency`	1 - 3600 segundos	Valor que especifica o intervalo de tempo de atualização da memória cache
`ConfigFrequency`	1 - 3600 segundos	Descreve o tamanho da memória cache respetiva ao armazenamento do histórico de dados
`DataSenderFrequency`	1 - 3600 segundos	Define o valor da memória cache respetiva à indexação da frequência e envio de dados referentes ao histórico
Zabbix-*proxy* em modo passivo		
`StartproxyPollers`	0-1000 instâncias	Define o número de processos de conexão, os quais devem ser aumentados consoante o número de proxys passivos
`proxyConfigFrequency`	1 - 3600 segundos	Este parâmetro permite ajustar a periodicidade com que o servidor Zabbix envia a configuração
`proxyDataFrequency`	1 - 3600 segundos	Define o valor da memória cache respetiva à indexação da frequência e envio de dados referentes ao histórico

Todos os ajustes de desempenho seguiram a metodologia utilizada anteriormente na configuração do servidor Zabbix, desse modo foi empregue o "Template App Zabbix-Proxy" para monitorizar os servidores Zabbix-*proxy* (ver Anexo XXII).

Com recurso aos parâmetros da Tabela 31, respetivos ao modelo de funcionamento ativo, foi definido que os dados provenientes da monitorização seriam guardados durante duas horas na presença de uma possível falha de comunicação entre o Zabbix-*proxy* e o servidor Zabbix.

5.7.4 Métodos de monitorização adotados

A monitorização dos equipamentos de rede deverá ser realizada mediante as interfaces lógicas que o servidor Zabbix disponibiliza, nomeadamente:

- Zabbix agent
- JMX
- SNMP
- IPMI

Tendo em conta os requisitos da U. Porto, foram estudadas e implementadas duas formas de monitorização, concretamente a monitorização com recurso a agentes Zabbix instalados e configurados nos próprios servidores, bem como a monitorização tradicional dos nós de rede baseada no protocolo SNMP.

Monitorização de sistemas computacionais

O benefício em utilizar o conceito de agente é largamente superior ao uso do protocolo SNMP, no que concerne à monitorização de um dado servidor. Isto é, as métricas por definição e o uso de *Templates* genéricos do Zabbix cria um automatismo e um dinamismo superior na tarefa diária do processo de gestão e administração de rede. Com auxílio de um agente é possível definir diversos itens que monitorizam múltiplos serviços de um dado servidor.

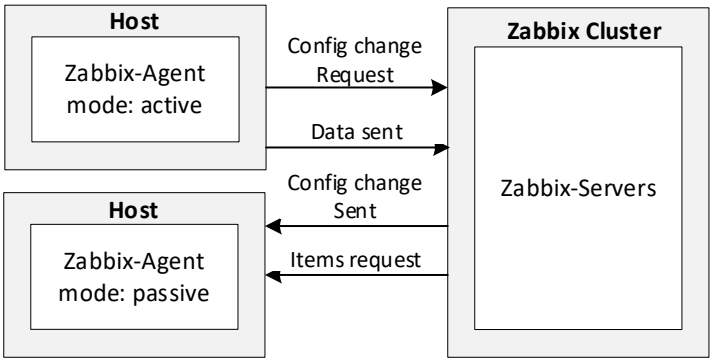


Figura 51 - Fluxo de dados em modo ativo e passivo nos agentes Zabbix

O modo de funcionamento do agente Zabbix é semelhante ao modelo do zabbix-*proxy* (Figura 51). O modelo passivo requer informação do servidor Zabbix para monitorar um dado *host*. Em contrapartida a monitorização nos agentes em modo ativo, aufere a independência do servidor Zabbix, sendo que o agente envia as configurações e os dados de monitorização ao

servidor Zabbix, sem que este faça qualquer tipo de pedido.

Embora a instalação dos agentes (ver Anexo XXIII) nos servidores que contemplam a solução atual não esteja em modo ativo, essa funcionalidade poderá de facto melhorar o desempenho do servidor Zabbix.

Com a utilização dos zabbix-*proxys*, a carga computacional já se encontra distribuída e, deste modo, a solução implementada foi desenhada a pensar nesse fator.

Com a utilização dos agentes Zabbix em modo passivo o Zabbix-*proxy* ganha mais relevo no que diz respeito às especificações de *hardware*. Por outro lado, consegue-se otimizar o desempenho de toda a solução de forma centralizada.

Posto isto, em casos em que o número de *hosts* monitorizados possa escalar e as especificações de *hardware* dos zabbix-*proxy* fiquem comprometidas, o uso de agentes Zabbix em modo ativo poderá ser a solução ideal.

Monitorização de equipamentos de rede

O processo de monitorização dos nós de rede foi projetado de acordo com as especificações dos equipamentos que decompõem a rede da U. Porto. Maioritariamente os dispositivos da rede de núcleo e distribuição permitem a utilização do protocolo SNMP (versão 2), o que se adequa à arquitetura do Zabbix, pois esta contém inúmeros *Templates* pré-definidos que se coadunam e oferecem *checks* aos serviços comumente utilizados.

O protocolo SNMP usufrui de indentificadores de objetos (OID), estando estes presentes na base de dados de gestão de informação (MIB) para realizar o método de monitorização local de um dado dispositivo.

Posto isto, analisou-se o número total de equipamentos de rede (Tabela 8) e constatou-se que a grande maioria dos *switches*, *routers*, *firewalls* e controladoras *wireless* são da marca Cisco. Compreendeu-se de igual modo, que os equipamentos de resiliência energética (UPSs) são suportados por dois fabricantes apenas, entre os quais a APC (*American Power Conversion*) e a Riello.

A tradução dos OIDs pode então ser indexada a MIBs específicas que cada fabricante disponibiliza para instalar nos servidores de monitorização.

No caso dos servidores Zabbix, foi necessário parametrizar e fazer o *download* das MIBs proprietárias (Anexo XXIV). Isto não só incrementou uma interoperabilidade entre os OIDs que o servidor passou a conseguir mapear, baseando-se única e exclusivamente na introdução (em ambiente gráfico) do nome da MIB no campo "`SNMP OID`", bem como, garantiu ainda uma monitorização dinâmica dos serviços, cobrindo assim os cenários de modelos de equipamentos distintos.

Através da comunidade do Zabbix [79], foi possível encontrar *templates* fidedignos, os quais a aplicação Zabbix por definição não contém. De acordo com a organização, todos os *Templates* foram definidos e restruturados mediante os equipamentos e os serviços a monitorizar (Tabela 32).

Tabela 32 - Templates utilizados para monitorizar equipamentos de rede

Equipamentos	Marca/Modelo	Templates genéricos (reajustados)
Routers	Cisco ASR-9010	Template Net Cisco IOS SNMPv2 ASR9000
	Cisco ASR-9006	
Switches	Cisco 2950	Template Net Cisco IOS SNMPv2
	Cisco 2960	
	Cisco 3560	
	Cisco 4500	
	Cisco 3600x ME	
	Cisco Nexus 5548	
	Cisco Nexus 2248 FEX	
	Cisco Nexus 2348 FEX	
	Cisco Nexus 9000	
Firewalls	Cisco ASA 55xx	Template Cisco ASA Discovery
Controladoras *wireless*	Cisco 8510 / 2504	Template Cisco WLC Discovery
UPS	APC / Riello	Template UPS TRIPH-APC Template UPS TRIPH-Riello

Cabe salientar, que a priorização dos *checks* aos serviços monitorizados foi configurada com um valor inferior a dois minutos, a fim de diminuir ao máximo o tráfego de gestão da rede, bem como para não subcarregar os equipamentos de pedidos ICMP (verificação de disponibilidade), salvaguardando o desempenho geral da solução.

5.7.5 Descoberta automática de equipamentos e serviços

O estudo da rede de gestão passou por avaliar a topologia lógica, de modo a compreender a melhor forma de integrar a solução.

A segmentação da rede de gestão de equipamentos ajudou ao entendimento hierárquico da rede de núcleo e distribuição (Figura 22), isto é, os nós de rede (*routers*) dão suporte às *defualt gateway* das redes de gestão de configurações (camada 3 do modelo OSI) dos *switches* e, as *firewalls* segregam os múltiplos domínios *broadcast* referentes às redes de gestão dos servidores (gestão Dom0, *Control Domain*, e ILO, *Integrated Lights Out*).

Do ponto de vista teórico, a rede de acesso (Figura 22) pode ser fundamentada pelos *switches* ME presentes em cada UO (faculdades, entre outros), bem como os *switches* de cada *datacenter* presentes nos pólos 1, 2 e 3. Por outro lado, ao nível da topologia física (camada 1 do modelo OSI) os sistemas computacionais centrais encontram-se interligados a *switches* e/ou, diretamente ligados aos *routers* ASR.

Tabela 33 - Principais redes utilizadas para autodescoberta de *hosts*

Descrição da rede	Locação	Endereço IP de rede	Protocolo de descoberta (*hosts*)
Equipamentos de rede			
Redes de gestão de *datacenters*	Pólo 1	172.17.10.0/24	Agente SNMPv1 e SNMPv2
		172.17.11.0/24	
	Pólo 2	172.17.20.0/24	
	Pólo 3	172.17.30.0/24	
Rede de gestão de equipamentos de núcleo distribuição e acesso		172.18.0.0/24	
Sistemas computacionais			
Redes de gestão (ILO)	Pólo 1	172.17.14.0/24	Agente zabbix
	Pólo 2	172.17.24.0/24	
	Pólo 3	172.17.34.0/24	
Redes de gestão (Dom0)	Pólo 1	172.17. 8.0/24	
	Pólo 2	172.17.28.0/24	
	Pólo 3	172.17.38.0/24	

Com esta análise compreendeu-se ainda que o número de equipamentos a monitorizar é elevado, o que revela uma importante mais-valia na adoção do modelo de descoberta de baixo nível (*Low-level Discovery*, LLD).

A metodologia de LLD do Zabbix possibilita a descoberta automática baseada nos seguintes métodos:

- Interfaces de rede (agente Zabbix);
- Ficheiros de sistema (agente Zabbix);
- Valores de CPU (agente Zabbix);
- Tabelas SNMP;
- *Queries* ODBC;
- LLD personalizado

Deste modo os endereços IP das redes de gestão (Tabela 33) foram analisados com recurso ao sistema LLD do Zabbix, definindo como métodos de descoberta o protocolo SNMP e o agente Zabbix (ver Anexo XXV).

A descoberta de baixo nível de equipamentos de rede e sistemas informáticos dependeu da parametrização de dois critérios fundamentais:

- A definição dos atributos protocolares respetivos ao processo de descoberta (Figura 106);
- A configuração de ações para que os *hosts* sejam adicionados dinamicamente aos *Templates*, grupos e inventários pretendidos (Figura 107).

Em relação aos serviços críticos de rede (*items*), o número é obviamente superior ao dos equipamentos, originando de igual modo, a pertinência em automatizar a sua descoberta, acautelando assim a componente de monitorização, tanto na vertente alarmística como na vertente estatística.

A arquitetura da descoberta de serviços (Figura 108 e Figura 110) compreendeu os seguintes conceitos elementares:

- Uma regra de descoberta depende da associação a um *template* de monitorização previamente configurado;
- Numa fase inicial, é definida a macro chave de descoberta, que irá identificar todos os *items* (serviços) descobertos (exemplo: descrição/nome das interfaces de rede).
- Na segunda fase de configuração, o conceito de “Item”, ganha o nome de “Item Prototype”, funcionando como uma finalização do processo de descoberta específico

ao que é pretendido monitorizar (exemplo: tráfego de entrada das interfaces de rede).

- Podem ser ainda definidos vários protótipos para além dos itens, como por exemplo, *triggers* (notificações e alertas) e/ou gráficos (monitorização estatística) que poderão ser criados dinamicamente mediante as informações das regras de descoberta anteriormente definidas.

5.8 Proposta de melhoria da topologia de monitorização

O cenário que se encontra em produção, de modo geral, seguiu uma metodologia de monitorização distribuída (ver Figura 52).

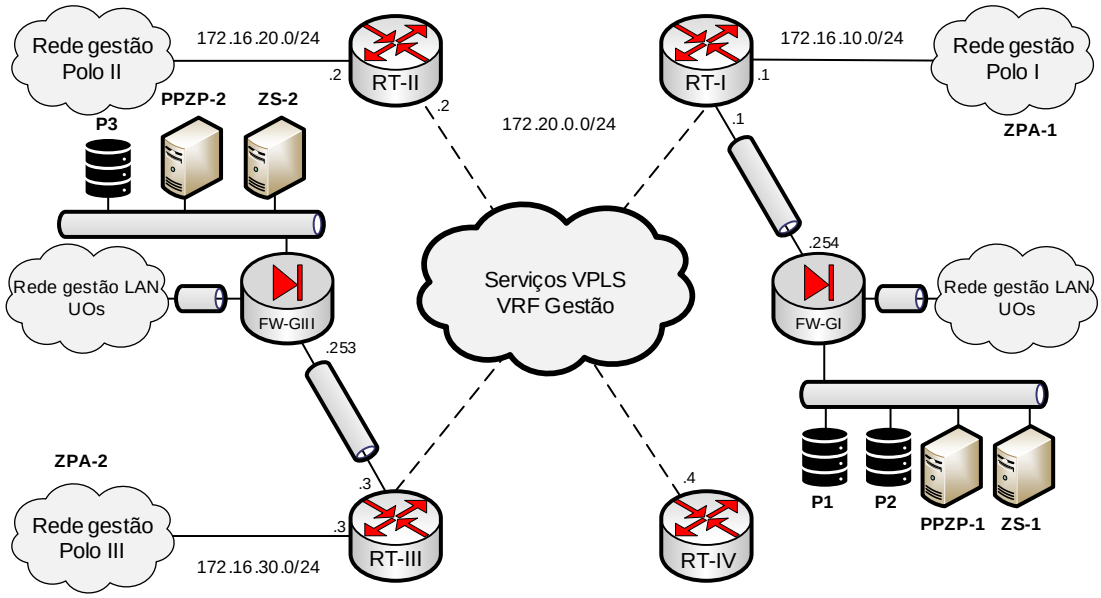


Figura 52 - Arquitetura de monitorização implementada

Todos os dispositivos monitorizados fora da rede local ao Zabbix, foram adicionados aos *clusters* de *proxys* (*proxy* Percona *and* Zabbix-*proxy*, PPZP), onde se encontram instalados os *softwares* proxySQL, HAproxy e Zabbix-*proxy*.

Com a proposta para a implementação final (Figura 53), sugere-se que a rede de gestão seja utilizada para o processo de monitorização alarmística e estatística, contemplando no mínimo dois *clusters* de servidores Zabbix-*proxy*, completamente dissociados dos proxys Percona (PP) melhorando as necessidades de segregação de tráfego, assim como, a segmentação hierárquica da rede atual.

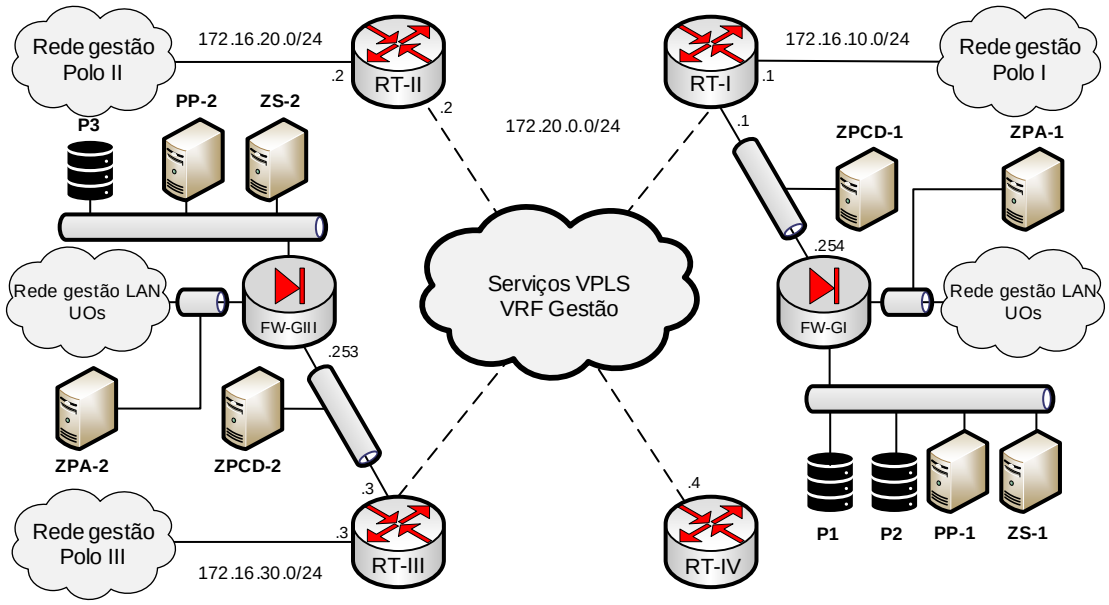


Figura 53 - Arquitetura de monitorização proposta

Particularmente, esta segmentação deverá sucumbir-se de um *cluster* de ZPCD (*Zabbix-proxy Core and Distribution*) inserido na rede de gestão alusiva ao núcleo e distribuição (172.18.0.0/24), a qual é totalmente transversal (mecanismos de MPLS e *routing*) às redes de gestão dos *datacenters* (VRF de gestão).

Propôs-se ainda, a criação de uma nova rede de gestão (Gestão LAN Zabbix), que aglomere o ZPA (*Zabbix-proxy Access*), de modo a este sistema ser completamente dedicado à monitorização de dispositivos das unidades organizacionais da U. Porto.

As bases de dados (*Percona*, P), e os sistemas computacionais ZS (*Zabbix Servers*), pertencem à rede de *Logs* (já existente), sendo esta completamente isolada dos sistemas de *proxy* referidos.

5.9 Integração da solução na infraestrutura de rede de dados

A integração da solução na rede da U. Porto levou a cabo o estudo exaustivo de todas as tecnologias e protocolos existentes, tendo como objetivo fundamental, a compreensão do domínio da rede de gestão. Devido à complexidade e extensibilidade da rede, serão dados os passos genéricos no que respeita à configuração dos equipamentos intervenientes.

Para simplificar e enquadrar todos os passos de configuração, de forma estruturada e concisa, utilizou-se o modelo TCP/IP, o qual aufere uma arquitetura lógica segmentada.

Ao nível da camada de acesso à rede, os sistemas informáticos que decompõem a solução, foram virtualizados na infraestrutura do KVM. Este sistema é distribuído fisicamente pelos *datacenters* do Pólo 1 e do Pólo 3, garantindo assim a arquitetura de monitorização proposta, bem como todos os seus princípios de resiliência (Figura 53).

Ainda na camada de acesso à rede, enumera-se a ligação de dados, como uma das fases que requer maior atenção no que diz respeito à topologia da rede da U. Porto. A tecnologia MPLS foi requerida para transportar um novo serviço VPLS, sendo este instanciado através de um VLAN ID atribuído nos equipamentos de núcleo, distribuição e acesso, entre os quais:

- *Routers* (Cisco ASR-9010);
- *firewalls* de gestão (ASA 5545);
- *switches* de interligação de núcleo e distribuição (Cisco 3560);
- *switches* de Pólo respetivos à infraestrutura do KVM (Cisco 2950, Cisco 2960).

Relativamente à camada de rede do modelo TCP/IP, a definição do novo domínio *broadcast* inerente ao serviço VPLS anteriormente referenciado, necessita de alcançar os múltiplos domínios *broadcast* das redes de gestão LAN (já existentes).

O *routing* entre os domínios *broadcast* mencionados, depende então da configuração de *firewalls*, onde devem ser executadas as seguintes parametrizações:

- Criação de uma nova interface de rede na *firewall* de gestão (nomenclada, gestão LAN Zabbix e o VLAN ID atribuído) alusiva e totalmente dedicada ao *cluster* do ZPA;
- Definição de rotas nas *firewalls* das redes locais (Unidades Organizacionais), para que estes equipamentos consigam encaminhar o tráfego com destino à nova rede.

Sob a perspetiva da camada de transporte, todo o tráfego é analisado pelas múltiplas *firewalls* que constituem a rede. Para que a comunicação seja autorizada para os destinos dos equipamentos que pretendemos monitorizar, seguiu-se a seguinte lógica de configuração:

- Definição de *access-lists* com o IP de origem dos dois endereços VIP pertencentes aos *clusters* de Zabbix-*proxy* (ZPCD e ZPA), com o destino aos sistemas que era pretendido monitorizar;

- Na *access-list* respetiva, deve ser definida a porta 10051 (TCP) para que seja estabelecida a comunicação entre os servidores Zabbix e os *cluster* de Zabbix-*proxy*, como também, deve ser permitido o tráfego na porta 10050 (TCP), de modo a garantir a conectividade entre os *clusters* de Zabbix-*proxy* e os agentes Zabbix;
- Para os equipamentos de rede, os quais utilizam o protocolo SNMP, tornou-se necessário autorizar a porta 161 (UDP) na respetiva *access-list*;
- Procedeu-se à configuração de dois endereços de NAT (*static* NAT) para os dois *clusters* de Zabbix-*proxy* (ZPCD e ZPA), visto que existem sistemas computacionais que só têm endereços públicos para proceder à sua monitorização;
- Nos casos em que existiam túneis VPN *site-to-site* para interligar as *firewalls* de origem e destino (por inexistência de interligações físicas dedicadas), foi realizado o uso dos mesmos, acrescentando *access-lists* análogas aos túneis, bem como procedendo à configuração de mecanismos de NAT zero ou NAT estático (versão 8.4 ou superior).

No ponto de vista da camada aplicacional, o presente relatório retrata todas as configurações a esse nível, dividindo cada componente integrante da solução de monitorização apresentada. Salienta-se que as comunidades SNMP (versão 1 e 2) foram configuradas em todos os equipamentos de rede, bem como foi permitido o acesso local (parametrizado no próprio equipamento) dos endereços VIP de origem, específicos a cada *cluster* de Zabbix-*proxy*.

Capítulo 6

6 Conclusão

6.1 Considerações finais

A Universidade do Porto dispõe de uma infraestrutura de comunicações que serve 51 Unidades Orgânicas, às quais oferece serviços de conectividade internos e externos. Transversalmente, o procedimento de administração e controlo da rede depende da interação com diferentes plataformas que providenciam a monitorização de, aproximadamente, quinhentos e noventa e cinco equipamentos ativos e mil setecentos e noventa e três serviços críticos de rede.

Contudo, a ausência de uma infraestrutura de monitorização centralizada obriga a múltiplos desafios operacionais, os quais, do ponto de vista da gestão e manutenção de rede, traduzem-se no aumento de inatividade desta, em perdas de produtividade, no défice temporal para a exploração de novas tecnologias automatizadas e, por conseguinte, na diminuição da qualidade de serviço prestada à comunidade académica.

Para sustentar a base teórica que deu suporte à implementação da solução, foi efetuada uma revisão da literatura, tendo como princípios orientadores o entendimento de arquiteturas basilares (OSI e TCP/IP) que apoiam abordagens mais simples e pragmáticas dos modelos e paradigmas da gestão de redes. Neste contexto, percebeu-se a importância do modelo de comunicação SNMP, pois este continua a ser a melhor opção para alicerçar o processo de monitorização dos equipamentos de rede, uma vez que a maioria dos sistemas em vigor não dão suporte às estruturas de referência mais modernas, designadamente, à gestão de redes baseada em *web* ou em políticas.

A realização do estudo e caracterização da topologia de rede física e lógica da U. Porto providenciou a visão da dimensão da rede, bem como colmatou o entendimento protocolar (IP, OSPF e BGP) e das tecnologias (MPLS e VPLS), que originam os fluxos de tráfego ocorridos nos equipamentos de comunicação. Deste modo, conclui-se que a tecnologia de transporte IP/MPLS é o princípio crucial para ter em consideração na integração e conceção de um sistema computacional dedicado à gestão e monitorização centralizada, partindo sempre do pressuposto que a metodologia de resiliência é disponibilizada pelos serviços VPLS.

A identificação e caracterização das tarefas de gestão realizadas pela Unidade de Serviços de Rede Centrais e *Datacenters* foi avaliada sob as diretivas propostas pelo modelo FCAPS. A análise comparativa efetuada mitigou e avaliou as lacunas das ferramentas de gestão de redes num dos ambientes virtualizados existentes (OpenStack), destacando a escolha da aplicação Zabbix como a plataforma que melhor se adequa às necessidades de gestão de falhas e gestão de desempenho. Por outro lado, para a plataforma de gestão de configurações recomenda-se a aplicação Oxidized. Destaca-se ainda, o *software* Graylog como sendo aquele que cumpre eficazmente os critérios estabelecidos para a gestão de contabilização e segurança.

Relativamente à infraestrutura de monitorização implementada, a mesma encontra-se em operação. Ao nível estrutural, a solução apoia-se na interação de sete sistemas informáticos, estando estes segmentados em três componentes computacionais. Cada elemento, está integrado numa arquitetura de *clusters* com diferentes funções, mormente: o armazenamento de dados (três nós Percona XtraDB *cluster*), o encaminhamento de pacotes (dois servidores ProxySQL) e a consulta de métricas de gestão para a monitorização dos dispositivos (dois sistemas Zabbix).

Em suma, é possível afirmar que os objetivos deste projeto de mestrado foram alcançados, na medida em que, foi possível desenvolver uma solução centralizada que cumpre com os requisitos estipulados pela U. Porto. Em seguida, importa ainda dissecar os resultados obtidos durante a fase de experimentação, conceção e finalização do sistema de gestão que apoia a monitorização.

6.2 Análise dos resultados

Durante o desenvolvimento da nova infraestrutura de monitorização foram sentidas diversas dificuldades, as quais fundamentam uma análise cuidada dos resultados. Tendo como base o estudo da documentação e da observação comportamental de cada sistema computacional, mediante as configurações que iam sendo realizadas, chegou-se ao entendimento de critérios essenciais.

As novas plataformas de gestão recorrem a modelos e paradigmas de base de dados relacionais, levando deste modo a concluir que as linguagens SGBD, Sistema de Gestão de Base de Dados, foram escolhidas tendo em conta a disponibilidade de dados que o sistema necessita, isto é, para transações diretas adequa-se o MySQL (eleito) e, por sua vez, o PostgreSQL consegue corresponder melhor às expetativas dos sistemas que dependem de consultas mais complexas.

As derivantes da linguagem nativa MySQL, levaram à comparação de soluções *open-source*. Pressupondo este facto, o sistema de gestão de base de dados MariaDB mostrou-se menos capaz por não oferecer uma topologia de encaminhamento resiliente sem recorrer a modelos pagos (MaxScale), adequando-se melhor a aplicação Percona que oferece uma arquitetura de *proxys* (ProxySQL) gratuita, que encaminha os dados ao nível da camada aplicacional.

No que concerne à resiliência da informação alusiva ao armazenamento, foi seguida a metodologia "multi-mestre", a qual define a sincronização da mesma informação aos três nós elencados ao *cluster* de base de dados. Não só por esta enorme vantagem, decidiu-se colocar estes sistemas autónomos na sua arquitetura de resiliência, sendo que com a metodologia "mestre-*backup*", existe maior probabilidade de disrupção dos dados, salientando que o Percona não suporta essa funcionalidade.

A componente de armazenamento demonstrou a enorme eficácia na utilização do Galera-c*luster* (nativo ao Percona), no que respeita à sincronização dos dados de armazenamento entre os nós. Mostrou-se também extremamente rápida e completamente transparente do ponto de vista do utilizador. Em caso de falha de um, dois, ou três sistemas de armazenamento, constatou-se que o mecanismo `xtrabackup` é superior ao `mysqldump` e `rsync,` pela sua fiabilidade e rapidez ao ressincronizar as configurações num cenário de catástrofe.

Para além disto, as restantes componentes (servidores de encaminhamento e servidores de consulta de dados), reúnem os princípios de redundância através da ferramenta Pacemaker. Numa primeira abordagem percebeu-se que a aplicação Keepalive não garante a recuperação com base nos processos da máquina, ou seja, se um dos processos parar, o sistema continua a responder ao endereço VIP em questão. Paralelamente, detetou-se outra menos valia, uma vez que o conceito de prioridades dos processos aplicacionais não é respeitado pela ordem correta, sendo que o Pacemaker recorre a configurações do *Corosync* para poder fazer a gestão apropriada dos processos.

Relativamente à componente de consulta de dados, o Zabbix revelou alguns problemas no armazenamento dos mesmos, gerando a inconsistência destes, nos nós do Percona. Tal como referido anteriormente, a metodologia "multi-mestre" foi a escolhida, levantando assim a premissa de que todas as tabelas existentes neste modo de funcionamento precisam de um campo primário (*primary key*), de modo a identificar exclusivamente cada registo. A resolução desta problemática passou por recriar o *script* de conceção da base de dados do Zabbix,

adicionando os respetivos campos primários.

Em relação ao desempenho da solução, foi reajustado o motor de base dados InnoDB, tendo em conta os valores atuais monitorizados pelo Zabbix, estando este a dar suporte a vinte e nove mil trezentos e quarenta e seis itens, oitenta e um terminais, cento e cinquenta e um *templates* e onze mil trezentos e cinco *triggers*.

Ainda sobre o conceito de performance, recai outro ponto de vista intrínseco à arquitetura projetada para o sistema Zabbix. Posto isto, avaliou-se a ação comportamental dos sistemas de Zabbix-*proxys*. No primeiro cenário definiu-se uma base de dados de armazenamento (MySQL) comum aos dois sistemas computacionais. Rapidamente se entendeu a enorme latência na consulta à informação dados, como também, surgiram múltiplos erros de inconsistências, pois um dado Zabbix-*proxy* deixa de ter conhecimento das alterações realizadas por outro sistema com a mesma função. De seguida, no segundo cenário recorreu-se ao SQlite, configurando-se este localmente em cada Zabbix-*proxy*, resolvendo não só a insistência dos dados armazenados, tal como ainda os atrasos às solicitações da informação de gestão.

O tipo de operação do Zabbix-*proxy* escolhido deve atender aos dois modos de atividade existentes (ativo e passivo), caracterizando-se a superioridade detetada para o modo de funcionamento ativo, o qual se traduziu na distribuição da carga computacional do servidor Zabbix.

No que respeita às mais-valias de automação no processo de gestão de falhas e desempenho, evidencia-se a utilização dos mecanismos de autodescoberta e LLDs do Zabbix, provendo a infraestrutura de melhoramentos na atividade de configuração, adição e de monitorização de dispositivos, deixando esta de estar dependente da ação manual da equipa técnica.

Para quantificar e correlacionar diferentes métricas com a utilização de expressões regulares, integrou-se a ferramenta Grafana (*frontend*) na infraestrutura centralizada baseada em Zabbix (*backend*), obtendo assim um refinamento de toda a informação extraída, aumentando a proficiência dos dados estatísticos obtidos.

Perante a impossibilidade de dar término às mais múltiplas tarefas de aprimoramento para o sistema desenvolvido e, tendo em conta o âmbito de estudo deste projeto, foi desenvolvida uma secção de trabalho futuro onde, em seguida, serão expostas um conjunto de medidas, retratando estas as vantagens competitivas e estratégicas para dar continuidade à evolução positiva da gestão operacional e da rede de gestão.

6.3 Trabalho futuro

Enumeram-se de seguida um conjunto de ações em termos de trabalho futuro, a realizar no âmbito do projeto em operação e enquadrado pelo presente relatório.

Com o objetivo de melhorar a infraestrutura de monitorização implementada e a gestão operacional, foram definidas e identificadas algumas tarefas que, a curto e médio prazo, poderão ser desenvolvidas, nomeadamente:

- Remover a aplicação HAProxy, utilizando diretamente os ProxySQL por forma a encaminhar os dados para o *cluster* de armazenamento;
- A solução em operação deve ser reestruturada de acordo com a proposta de melhoria da topologia de monitorização (Figura 52- em operação; Figura 53 - proposta);
- Em relação ao motor de consulta de dados adotado, o Zabbix apenas contempla a monitorização dos dispositivos de rede e sistemas informáticos mais críticos. Existe a necessidade de adicionar os restantes equipamentos e serviços da rede da U. Porto ao sistema de monitorização em vigor;
- Criar um *template* em Zabbix que providencie a monitorização estatística de serviços de rede IP/MPLS ao invés de interfaces físicas dos equipamentos. Deste modo, seria possível obter informação específica do volume de tráfego utilizado por cada serviço (VLAN ID);
- Integrar a plataforma Zabbix com os servidores LDAP centrais da U. Porto de modo a centralizar a autenticação.;
- No que concerne à resiliência da infraestrutura de monitorização em operação salienta-se que foram detetados, esporadicamente, problemas de *split-brain* nos sistemas de *cluster* que só possuíam dois nós de computação. Como tal, sugere-se o incremento de um novo sistema operacional que ofereça suporte à aplicação Pacemaker interligando-a a todos os *clusters* que se encontrem neste contexto;
- Descontinuar ou rever a utilidade das ferramentas Nagios e RRDTool/Weathermap;
- Reavaliar a necessidade da aplicação Syslog-NG revendo a arquitetura dos sistemas de *logs* por forma a possibilitar a receção dos registos dos dispositivos de rede diretamente na plataforma Graylog;

- Definir na aplicação Zabbix o conceito *profiling* em múltiplos ambientes, de forma a segmentar o acesso pelos utilizadores de cada Unidade da U. Porto;
- Estudar um mecanismo de resiliência para a ferramenta de gestão de configurações Oxidized;
- Elencar todas as plataformas inerentes do modelo FCAPS ao *cluster* de armazenamento implementado.

De igual modo, é também possível identificar um conjunto de tarefas que, a longo prazo, poderão vir a ser desenvolvidas de modo a acrescentar valor, quer à rede de gestão da U. Porto, quer à nova solução concretizada durante a elaboração deste projeto. Em detalhe apresentam-se os seguintes pontos de ação:

- Atualizar o modelo de gestão alusivo à comunicação dos dispositivos de rede, para o protocolo SNMPv3. Nos casos em que não for possível, propõe-se a aquisição de novos sistemas de rede;
- Mediante os novos modelos e paradigmas para gestão de redes, os modelos de comunicação NETCONF (Anexo II) e YANG irão, futuramente, substituir o protocolo SNMP. Assim, justifica-se que a U. Porto reestruture a sua infraestrutura de rede, de modo a provir este protocolo e esta linguagem de modulação de dados;
- Implementação de um meio de comunicação diferenciado baseado na gestão de rede *out-of-band* de modo a separar fisicamente o tráfego de gestão, providenciando uma forma de aceder aos equipamentos em caso de indisponibilidades de rede;
- Na hipótese de a U. Porto reestruturar a sua infraestrutura de comunicação de dados para uma nova abordagem assente na arquitetura de rede SDN, a ferramenta Prometheus deve ser considerada de acordo com os resultados obtidos neste projeto;
- Automação das configurações, especificamente, dotar o ambiente relativo à gestão de rede, de uma plataforma que uniformize as parametrizações diárias dos equipamentos de núcleo, de distribuição e de acesso;
- Prevê-se que mediante o incremento de grandes volumes de dados provenientes da monitorização seja necessário ter de substituir a linguagem de consulta de base de dados para PostgreSQL devido a esta corresponder de melhor forma a consultas mais complexas;

- Explorar e mitigar as necessidades para atualizar a nova infraestrutura de monitorização, tendo em conta os seguintes conceitos tecnológicos: *container*, *dockers*, *kubernetes* e micro serviços;
- Implementação de um NOC (*Network Operations Center*) de âmbito geral à U. Porto, de modo a permitir uma gestão eficaz de todos os recursos.

Referências

[1] A. L. Cervo and P. A. Bervian, “Metodologia Científica.” São Pauo, 2002.

[2] J. Rosgen, B. M. Pettitt, and D. W. Bolen, *“Data and Computer Communications,”* 10th ed., vol. 16, no. 4. 2007.

[3] E. Monteiro and F. Boavida, *“Engenharia de Redes Informáticas,”* 10th ed. 2011.

[4] D. Plummer, “Ethernet Address Resolution Protocol Or Converting Network Protocol Addresses,” 1982.

[5] J. Postel, “RFC 792 Internet Control Message Protocol,” 1981.

[6] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, “RFC 3376 Internet Group Management Protocol, Version 3,” 2002.

[7] J. Postel, “RFC 768 User Datagram Protocol,” 1980.

[8] B. Soediono, “RFC 793 TRANSMISSION CONTROL PROTOCOL DARPA INTERNET PROGRAM,” 1981.

[9] M. Subramanian, T. A. Gonsalves, and N. U. Rani, *“Network management: Principles and Practice.”* 2010.

[10] U. S. Warrier and L. Besaw, “RFC 1095, ‘O Common Management Information Services e o Protocolo sobre TCP / IP (CMOT),’” pp. 1–67, 1989.

[11] S. A. Heinz-Gerd Hegering and B. Neumair, *“Integrated management of networked systems : concepts, architectures, and their operational.”* 1999.

[12] W. Stallings, *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2*, Addison-We. 1998.

[13] J. . Davin, J. Case, M. Fedor;, and M. Schoffstall, “RFC 1028 ‘A Simple Gateway Monitoring Protocol,’” 1987.

[14] M. T. Rose; and K. McCloghrie, “RFC 1155 ‘Structure and Identification of Management Information for TCP/IP-based Internets,’” p. 21, 1990.

[15] K. McCloghrie, D. Perkins, and J. Schoenwaelder;, “RFC 2579 ‘Textual Conventions

for SMIv2,’” p. 26, 1999.

[16] J. Case, M. Fedor, M. Schoffstall, and J. Davin, “RFC 1157, ‘A Simple Network Management Protocol (SNMP),’” pp. 1–36, 1990.

[17] J. D. Case, K. McCloghrie, M. T. Rose, and S. Waldbusser, “RFC 1448 ‘Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2),’” p. 35, 1993.

[18] J. Case, R. Mundy, D. Partain, and B. Stewart, “RFC 3410 Introduction and Applicability Statements for Internet Standard Management Framework,” 2002.

[19] International Organization for Standardization, “Information Technology - Open Systems Interconnection - Management Information Services - Structure of Management Information,” 1993.

[20] K. McCloghrie and M. Rose, “RFC 1213, ‘Management Information Base for Network Management of TCP/IP-based internets: MIB-II,’” pp. 1–70, 2011.

[21] S. Waldbusser, “RFC 2819, ‘Remote Network Monitoring Management Information Base,’” vol. 1, no. 20, pp. 1–98, 2000.

[22] S. Waldbusser, “RFC 4502 Remote Network Monitoring Management Information Base Version 2,” 2006.

[23] S. Waldbusser, “RFC 2021 Remote Network Monitoring Management Information Base Version using SMIv2,” 1997.

[24] International Telecommunication Union, “M.3400 : TMN management functions,” *International Telecommunication Union*, 2000. .

[25] Flextronics Software System, “FCAPS White Paper,” 2005.

[26] P. Goyal, R. Mikkilineni, and M. Ganti, “FCAPS in the business services fabric model,” in *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, 2009.

[27] M. Jude, “‘Network Management Tools And Trends,’” *Business Communications Review*, vol. 32, no. 5. p. 50, 2002.

[28] J. Pires, “Gestão técnica e operacional da rede metropolitana da Associação Porto Digital,” Instituto Universitário da Maia, 2018.

[29] I. Cisco, “Cisco SAFE Reference Guide,” no. 6387, 2009.

[30] Prometheus, “Prometheus | Introduction Overview.” [Online]. Available: https://prometheus.io/docs/introduction/overview/. [Accessed: 19-Sep-2019].

[31] T. Urban, *Cacti 0.8 Beginner’s Guide*. 2011.

[32] The Cacti Group, “Cacti Docs.” [Online]. Available: https://docs.cacti.net/. [Accessed: 10-Sep-2019].

[33] Tobias Oetiker, “RRDtool - About RRDtool.” [Online]. Available: https://oss.oetiker.ch/rrdtool/. [Accessed: 10-Sep-2019].

[34] T. C. Group, “Cacti® - The Complete RRDTool-based Graphing Solution.” [Online]. Available: https://www.cacti.net. [Accessed: 10-Sep-2019].

[35] A. D. V. and S. K. Lee, *Mastering Zabbix*, Second. 2015.

[36] Rihards Olups, *“Zabbix Network Monitoring - Second Edition.”* 2016.

[37] Zabbix SIA, “2 Item types [Zabbix Documentation 4.2].” [Online]. Available: https://www.zabbix.com/documentation/4.2/manual/config/items/itemtypes. [Accessed: 11-Sep-2019].

[38] A. Shokhin, “Network monitoring with Zabbix,” 2015.

[39] W. Kocjan, *Learning Nagios 4*. 2014.

[40] T. Ryder, *Nagios Core Administration cookbook*, Second Edi. 2016.

[41] M. Badger, *Zenoss Core 3.x Network and System Monitoring*. Packt Publishing, 2011.

[42] Zenoss Inc, “Zenoss Linux Monitor.” [Online]. Available: https://help.zenoss.com/in/zenpack-catalog/open-source/linux-monitor. [Accessed: 18-Sep-2019].

[43] Zenoss Inc, “Zenoss Microsoft Windows.” [Online]. Available: https://help.zenoss.com/in/zenpack-catalog/open-source/microsoft-windows. [Accessed: 18-Sep-2019].

[44] One Identity LLC, “Syslog-NG Open Source Edition Datasheet,” 2018.

[45] One Identity LLC, *The Syslog-NG Open Source Edition 3.8 Administrator Guide*. 2018.

[46] Graylog, “Graylog Enterprise Log Management.” [Online]. Available:

http://www.graylog.org/. [Accessed: 14-Sep-2019].

[47] Graylog, “Graylog Documentation.” [Online]. Available: http://docs.graylog.org/en/3.1/. [Accessed: 16-Sep-2019].

[48] Graylog, “Graylog | Bigger production setup.” [Online]. Available: https://docs.graylog.org/en/3.1/pages/architecture.html. [Accessed: 16-Sep-2019].

[49] Shrubbery Networks, “Shrubbery Networks, Inc - RANCID.” [Online]. Available: http://www.shrubbery.net/rancid/. [Accessed: 17-Sep-2019].

[50] S. Ytti, “Oxidized.” [Online]. Available: https://github.com/ytti/oxidized. [Accessed: 17-Sep-2019].

[51] S. Ytti, “Oxidized Web.” [Online]. Available: https://github.com/ytti/oxidized-web. [Accessed: 17-Sep-2019].

[52] J. Moy, “RFC 2328 : ‘OSPF Version 2,’” *RFC*, 1998.

[53] Y. Rekhter, T. Li, and S. Hares, “RFC 4271: A Border Gateway Protocol 4 (BGP-4),” 2006.

[54] E. Rosen, A. Viswanathan, and R. Callon, “RFC 3031 Multiprotocol Label Switching Architecture,” 2001.

[55] L. Andersson and E. Rosen, “RFC 4664 Framework for Layer 2 Virtual Private Networks (L2VPNs),” 2006.

[56] S. Frankel and S. Krishnan, “RFC 6071 IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap,” 2011.

[57] P. Calhoun, S. Farrell, G. Gross, and D. Spence, “RFC 2904 AAA Authorization Framework,” *RFC 2904*, pp. 1–35, 2000.

[58] J. Sermersheim, “RFC 4511 Lightweight Directory Access Protocol (LDAP): The Protocol,” pp. 1–69, 2006.

[59] H. Andrea, *Cisco ASA firewall fundamentals*. CISCO, 2011.

[60] R. Droms, “RFC 2131 - Dynamic Host Configuration Protocol.” 1997.

[61] Cisco, “Troubleshoot a Lightweight Access Point Not Joining a Wireless LAN Controller,” 2015. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/wireless/5500-series-wireless-

controllers/119286-lap-notjoin-wlc-tshoot.html. [Accessed: 11-Jul-2019].

[62] Cisco, “Lightweight AP (LAP) Registration to a Wireless LAN Controller (WLC),” 2019. [Online]. Available: https://www.cisco.com/c/en/us/support/docs/wireless-mobility/wireless-lan-wlan/70333-lap-registration.html. [Accessed: 11-Jul-2019].

[63] Prometheus, “Prometheus | Prometheus Storage.” [Online]. Available: https://prometheus.io/docs/prometheus/1.8/storage/. [Accessed: 22-Sep-2019].

[64] Zabbix SIA, “2 Requirements [Zabbix Documentation 4.4].” [Online]. Available: https://www.zabbix.com/documentation/4.4/manual/installation/requirements. [Accessed: 22-Sep-2019].

[65] Zenoss Own IT, “Zenoos Community Edition (Core).” Release 6.2.0, 2018.

[66] 2ndQuadrant, “Introducing PostgreSQL | PostgreSQL vs MySQL.” [Online]. Available: https://www.2ndquadrant.com/en/postgresql/postgresql-vs-mysql/. [Accessed: 17-May-2020].

[67] 2ndQuadrant, “PostgreSQL vs MySQL | Performance.” [Online]. Available: https://www.2ndquadrant.com/en/postgresql/postgresql-vs-mysql/. [Accessed: 17-May-2020].

[68] Microsoft Company, “Citus Data | Citus Documentation v9.2,” 2020. [Online]. Available: http://docs.citusdata.com/en/v9.2/. [Accessed: 17-May-2020].

[69] Percona, “Percona | MySQL vs. MariaDB: Reality Check,” 2017. [Online]. Available: https://www.percona.com/blog/2017/11/02/mysql-vs-mariadb-reality-check/. [Accessed: 17-May-2020].

[70] Zabbix SIA and V. Sokurenko, “ZABBIX BUGS AND ISSUES (ZBX-16465),” 2019. [Online]. Available: https://support.zabbix.com/browse/ZBX-16465. [Accessed: 17-May-2020].

[71] Zabbix SIA, “‘2 Requirements [Database size],’” 2019. [Online]. Available: https://www.zabbix.com/documentation/4.4/manual/installation/requirements. [Accessed: 07-Dec-2019].

[72] Kenny Gryp, “Technical Presentations | ‘Choosing Hardware for MySQL [Percona Live Washington DC’],” *Percona*, 2012. [Online]. Available: https://www.percona.com/resources/technical-presentations/choosing-hardware-mysql-

percona-live-washington-dc-2012. [Accessed: 10-Nov-2019].

[73] HAPROXY Community, “HAPROXY | Operating System and Hardware Requirements.” .

[74] Codership Ltd, “Galera Cluster | Database Replication,” 2014. [Online]. Available: https://galeracluster.com/library/documentation/tech-desc-introduction.html. [Accessed: 04-Aug-2019].

[75] Codership Ltd, “Galera Cluster | Replication API,” 2014. .

[76] Percona, “Percona | Multi-Master Replication,” 2017. [Online]. Available: https://www.percona.com/doc/percona-xtradb-cluster/LATEST/features/multimaster-replication.html. [Accessed: 12-Dec-2019].

[77] Percona, “Percona | PXC Strict Mode.” [Online]. Available: https://www.percona.com/doc/percona-xtradb-cluster/LATEST/features/pxc-strict-mode.html. [Accessed: 21-May-2020].

[78] Clusterlabs, “Pacemaker | 1.4 Pacemaker Architecture.” [Online]. Available: https://clusterlabs.org/pacemaker/doc/deprecated/en-US/Pacemaker/1.0/html/Pacemaker_Explained/s-intro-architecture.html. [Accessed: 20-Jan-2020].

[79] Zabbix SIA, “Zabbix Share | Network Devices,” 2020. [Online]. Available: https://share.zabbix.com/network_devices. [Accessed: 20-Apr-2020].

[80] Douglas R. Mauro and K. J. Schmidt, *“Essential SNMP - Help for System and Network Administrators,”* 2nd ed., vol. 136, no. 1. O’Reilly, 2007.

[81] W. Stallings, “‘SNMP and SNMPv2: The Infrastructure for Network Management,’” *IEEE Communications Magazine*, 1998.

[82] B. Wijnen, D. Harrington, and R. Presuhn, “RFC 3411 ‘An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks,’” pp. 1–64, 2002.

[83] M. Bjorklund, “RFC 6241 Network Configuration Protocol (NETCONF),” 2011.

[84] M. Wasserman, “Concepts and Requirements for XML Network Configuration,” 2002.

[85] M. Sloman, *Network and Distributed Systems Management*. 1994.

[86] G. Booch, I. Jacobson, and J. Rumbaugh, “The Unified Modeling Language Reference Manual,” in *Summary for Policymakers*, 1998.

[87] D. C. Verma, *“Principles of computer systems and network management,”* Springer. 2009.

[88] OMG, “The Common Object Request Broker: Architecture and Specification.” [Online]. Available: https://www.omg.org. [Accessed: 29-Jan-2019].

[89] T. J. Mowbray and R. Zahavi, *The Essential CORBA: Systems Integration Using Distributed Objects*. 1995.

[90] S. Mazumdar, “Inter-Domain Management between CORBA and SNMP : WEB-based Management - CORBA/SNMP Gateway Approach,” 1996.

[91] J. Strassner, *Policy-Based Network Management: Solutions for the Next Generation (The Morgan Kaufmann Series in Networking)*. 2003.

[92] J. Boyle, R. Cohen, S. Herzog, R. Rajan, and A. Sastry, “RFC 2748 The COPS (Common Open Policy Service) Protocol,” 2000.

[93] A. Westerinen *et al.*, “RFC 3198 Terminology for Policy-Based Management,” 2001.

[94] K. Chan *et al.*, “RFC 3084 COPS Usage for Policy Provisioning (COPS-PR),” 2001.

[95] Percona, “Percona | About Percona XtraBackup 2.4.” [Online]. Available: https://www.percona.com/doc/percona-xtrabackup/2.4/intro.html. [Accessed: 10-Feb-2020].

Anexos

Anexo I. Estudo do protocolo SNMP

Para a versão 1 do protocolo SNMP existem quatro operações que fazem sentido serem analisadas com maior detalhe [80]:

A operação *Get-Request* viabiliza que uma aplicação de gestão instalada num sistema NMS (gestor) leia valores da MIB de um MD (agente). O agente que recebe a mensagem *Get-Request* deve responder com uma mensagem *Get-Response* com o identificador pedido e com os valores solicitados. No caso de a aplicação falhar, a mensagem *Get-Response* irá conter um código de erro, de forma a identificar o tipo de erro ocorrido.

Caso exista a necessidade de ler vários objetos de uma só interação, ou quando não se conhece a extensibilidade da base de dados (MIB) existente no agente, é necessário utilizar a operação *Get-Next-Request*, o qual fornece o valor do objeto seguinte da MIB. Como resposta a uma mensagem do tipo *Get-Next-Request*, é recebida uma mensagem *Get-Response*.

Quando é pretendido que um agente coloque um ou mais objetos na MIB com os valores fornecidos, utiliza-se a operação *Set-Request*. Este é o tipo de operação de escrita, ao contrário das operações do tipo *Get*, que são apenas de leitura. O agente sinaliza com sucesso ou insucesso da operação através do envio de uma mensagem *Get-Response* ao gestor.

A operação *Trap* é utilizada sempre que um agente necessita de notificar um gestor de um acontecimento relevante. Este tipo de notificação, no protocolo SNMPv1, é assíncrona, isto é, não é solicitada pelo gestor. É uma situação excecional e é normalmente utilizada quando o agente deteta uma anomalia no dispositivo onde está configurado.

Um dispositivo gerido deve ter um agente conforme é ilustrado na Figura 54. Neste sistema de gestão é utilizado os protocolos SNMP, UDP e IP. A partir do sistema de gestão de rede (*Network Managed System*) são disponibilizados três tipos de operações: *Get-Request*, *Get-Next-Request* e *Set-Request*. As três operações são confirmadas pelo agente (*Network Managed Device*, NMD) na forma de uma notificação *Get-Response* que é devolvida ao sistema de gestão de rede. A comunicação (pedidos e respostas) entre o sistema de gestão e o agente SNMP decorre com recurso ao protocolo UDP na porta 161, e para a receção de *traps* na porta 162 [81].

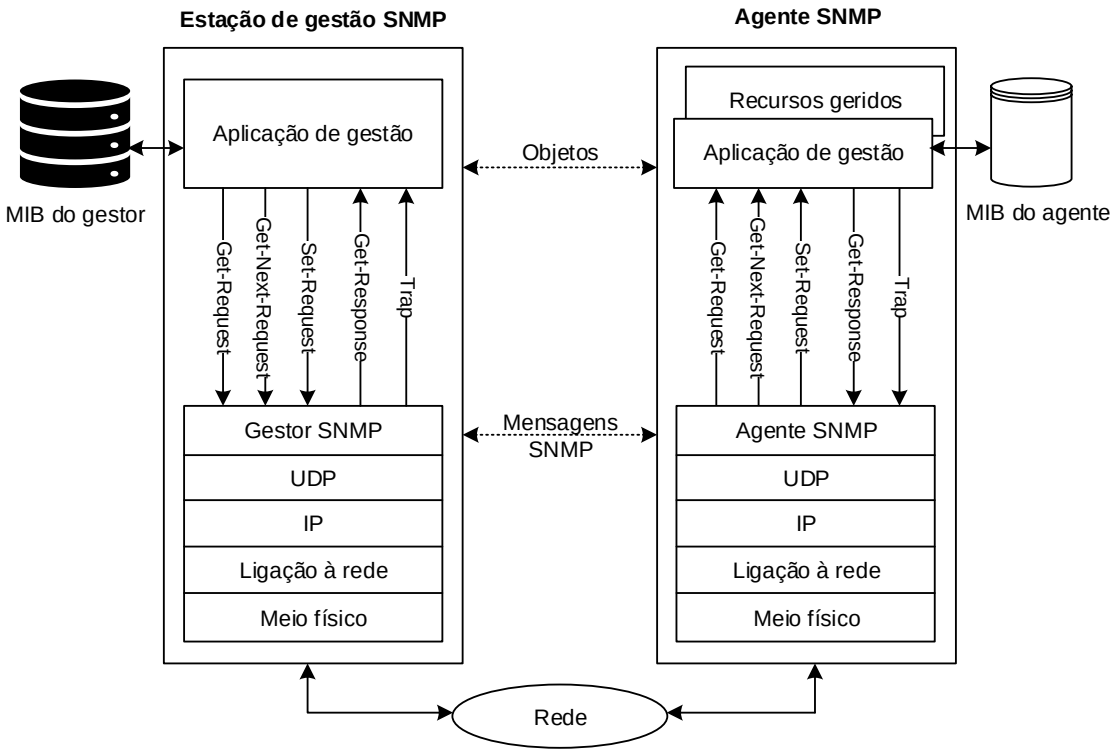


Figura 54 - Operações SNMPv1 na arquitetura TCP/IP [81]

Formato da mensagem do protocolo SNMP

No processo de comunicação entre o agente e o gestor, o protocolo SNMP troca mensagens transportadas num cabeçalho genérico (SNMP *message*). No cabeçalho SNMP, existem três campos principais: o campo que indica a versão (*Version*) utilizada pelo SNMP, o campo responsável por identificar o nome da comunidade (*Community Name*) SNMP, ao qual pertence o remetente da mensagem.

Por fim existe um campo de tamanho variável (SNMP PDU), que determina a unidade de dados da mensagem (*Protocol Data Unit*, PDU).

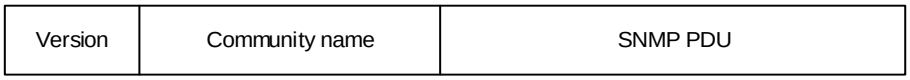


Figura 55 - Formato da mensagem SNMP

Formato da mensagem PDU-SNMPv1

O cabeçalho da unidade de dados contido na mensagem SNMP utiliza dois tipos de mensagens com diferentes campos conforme identifica a Figura 56.

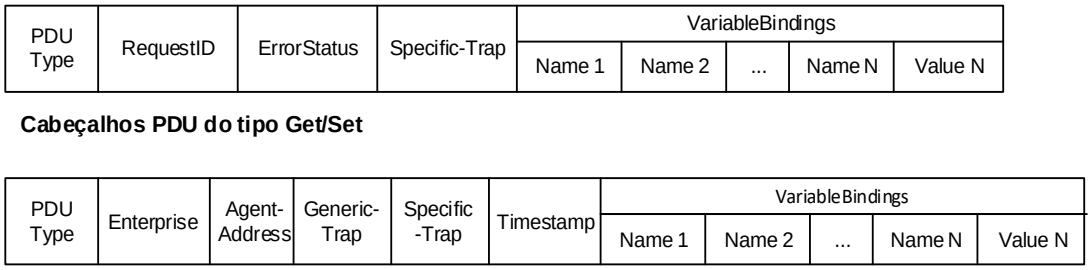


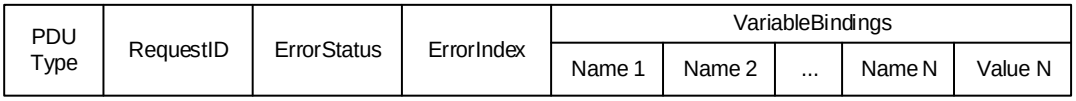
Figura 56 - Formato PDU SNMPv1

A mensagem PDU enviada no cabeçalho SNMPv1 utiliza as operações do tipo *Get* e *Set*, identificadas por um valor do tipo de PDU (entre 0 e 4). Por exemplo, o valor zero identifica que o PDU pertence a um cabeçalho proveniente de uma operação *Get-Request* [16]. Para controlar as respostas é utilizado o identificador *Request-ID*. Quando é realizado uma operação *Get-Response* é utilizado o campo *Error-Status* para indicar à entidade que emitiu o comando, o resultado, do seu pedido. Para auxiliar esta operação, o campo *Error-Index* funciona como um apontador, de forma a identificar o objeto que gerou o erro. O campo *Variable-Bindings* permite receber um conjunto de pares do tipo nome/valor, que identificam os objetos na MIB dentro do PDU.

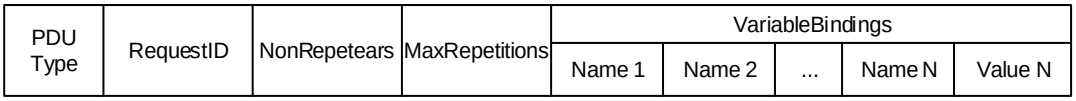
No cabeçalho de PDU-*Trap* é definido por alguns campos importantes que determinam o envio deste tipo de mensagens. O valor quatro, respetivo ao PDU-*Type*, determina o tipo de mensagem no formato *Trap*. Para identificar o dispositivo de rede que gerou a *Trap*, são utilizados os campos *Enterprise* e o *Agent-Address*. Existem valores genéricos (*Generic-Trap*) e específicos (*Specific-Trap*) de *Trap's*, que possibilitam a identificação da *Trap* segundo um conjunto de códigos predefinidos ou implementados.

Formato da mensagem PDU-SNMPv2

Para melhorar a eficiência e o desempenho no processo da troca de mensagens entre sistemas de gestão de redes, a estrutura de dados PDU referente ao protocolo SNMPv2, foi alterada conforme ilustra a Figura 57.



Cabeçalhos PDU genéricos



Cabeçalhos PDU do tipo Get-Bulk-Request

Figura 57 - Formato PDU SNMPv2

No cabeçalho PDU definido pela operação *Get-Bulk-Request*, o campo *Non-Repeaters* indica o número de objetos que não serão lidos repetitivamente. Por sua vez, o campo *Max-Repetitions*, designa o número máximo de interações que serão realizadas para os objetos (OID's) que se seguem. Este campo designa o número máximo de linhas (referente à tabela da MIB) que é solicitado pelo gestor [17].

Formato da mensagem SNMPv3

O SNMPv3 utiliza o mesmo formato de cabeçalho PDU, que o SNMPv2. No entanto, cada mensagem SNMPv3 inclui quatro grupos de dados adicionais: *msg-Version*, *msg-Globa-Data*, *msg-Security-Parameters* e *msg*-PDU, conforme é demostrado na Figura 58.

Para identificar a versão do SNMPv3, utiliza-se o campo *msg-Version* e um respetivo ID (preenchido com o valor 3). O campo *msg-Globa-Data* contém as informações do cabeçalho. Por sua vez, o campo *msg-Security-Parameters* é utilizado exclusivamente pelo *security model*, de forma a disponibilizar um conjunto de campos que possuem parâmetros, para serem posteriormente processados no subsistema de segurança. O *msg*-PDU é um campo que disponibiliza informações associadas um contexto exclusivo de um PDU.

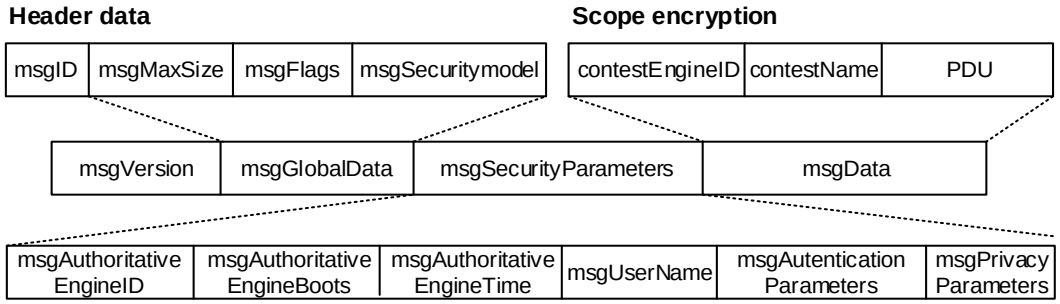


Figura 58 - Formato de mensagem SNMPv3

Na Figura 59, podemos observar o relacionamento entre os diferentes mecanismos de cada subsistema, numa arquitetura gestor-agente respetiva ao protocolo SNMPv3.

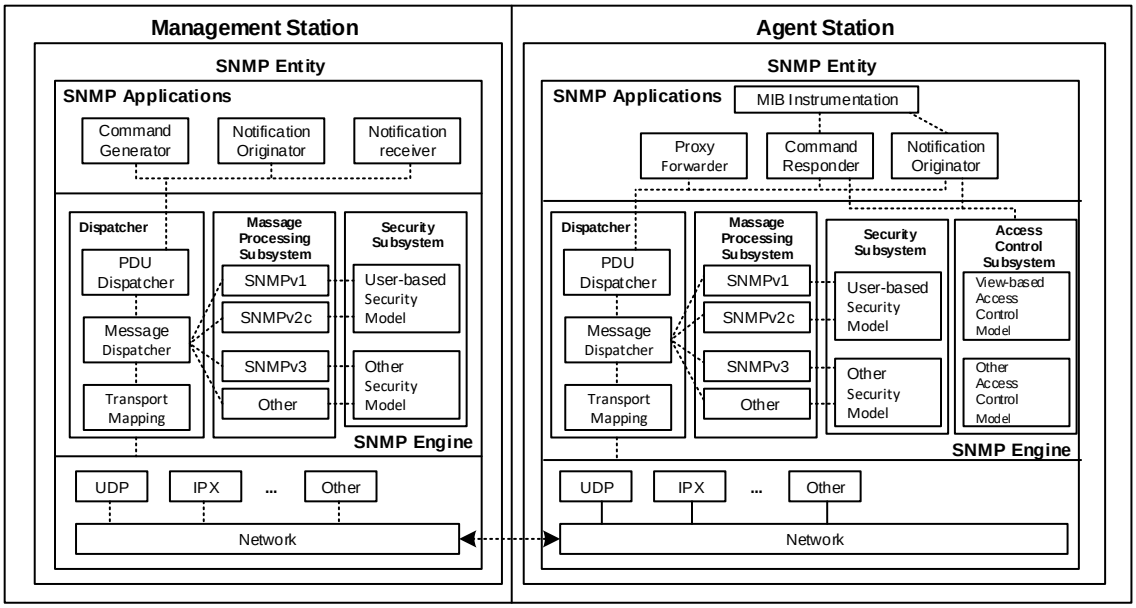


Figura 59 - Arquitetura do agente SNMPv3

As interfaces de serviço abstratas (*Abstract Service Interfaces*) representam as interfaces conceptuais entre os vários subsistemas, dentro de uma entidade SNMP. Estas são definidas por um conjunto de primitivas e elementos de dados abstratos. Uma primitiva especifica a função a ser executada e os parâmetros que devem ser utilizados para transmitir dados e controlar as informações. A Tabela 34 identifica a lista de primitivas que foram definidas para os diferentes subsistemas [82].

Tabela 34 - Lista de primitivas para elementos de dados abstratos

Componente	Primitivas	Serviço
Dispatcher	Send-PDU	Envia um pedido ou uma notificação SNMP para outra entidade SNMP
	Process-PDU	Comuta uma PDU de entrada para uma aplicação
	Return-Response-PDU	Retorna um PDU de resposta SNMP ao *Dispatcher*
	Process-Response-PDU	Comuta uma PDU de resposta SNMP para a entrada de uma aplicação
	Register-Context-Engine-ID	Regista um *contextEngineID* específico para um *pduTypes*
	Unregister-Context-Engine-ID	Não regista um *contextEngineID* específico para um *pduTypes*
Subsistema de Processamento de Mensagens	Prepare-Outgoing-Message	Cria um pedido SNMP de saída ou uma mensagem de notificação
	Prepare-Response-Message	Cria uma mensagem de resposta SNMP
	Prepare-Data-Elements	Prepara os elementos de dados abstratos a partir de uma mensagem SNMP recebida
Subsistema de Segurança	Generate-Request-Msg	Inicia um pedido ou uma mensagem de notificação
	Process-Incoming-Msg	Processa uma mensagem recebida
	Generate-Response-Msg	Gera uma mensagem de resposta
Subsistema de Controlo de Acesso	Is-Access-Allowed	Verificação de acesso
Modelo de Segurança baseado em Utilizadores (USM)	Authenticate-Outgoing-Msg	Autêntica uma mensagem de saída
	Authenticate-Incoming-Msg	Autêntica uma mensagem de entrada
	Encrypt-Data	Criptografa os dados
	Decrypt-Data	Desencripta os dados

O SNMPv3 define os procedimentos que devem ser seguidos para cada tipo de aplicação, tanto no processo de criação de PDU's, bem como no processamento ou transmissão destes. Os procedimentos são definidos em termos de interação com o *Dispatcher* de acordo com as suas primitivas.

Formalmente o protocolo SNMPv3 define cinco tipos de aplicações que utilizam serviços fornecidos pelos diferentes mecanismos [80]:

- Gestor de comandos (*Command Generator*) – estabelece pedidos do tipo *Get-Request*, *Get-Next-Request*, *Get-Bulk* e *Set-Request*, e processa as respostas. Esta aplicação é implementada por um *Network Management System* (gestor), que pode emitir e consultar solicitações do tipo *Set*, para as diferentes entidades, presentes em *routers*, *switches* entre outros;

- Comandos de resposta (*Command Responder*) – responde aos pedidos dos comandos *Get-Request*, *Get-Next-Request*, *Get-Bulk* e *Set-Request*, através das primitivas do módulo *Dispatcher* (*register-Context-Engine-ID, unregister-Context-Engine-ID, process*-PDU *e return-Response*-PDU) e do Subsistema de Controlo de Acessos (*is-Access-Allowed*). Posteriormente o módulo *Dispatcher* comuta cada PDU de entrada para a respetiva aplicação com recurso à primitiva *process-PDU*. Por fim, o gerador de comandos (*Command Generator*) utiliza a primitiva *return-Response*-PDU, de forma a retornar um PDU de resposta ao *Dispatcher*;

- Gestor de notificações (*Notification Originators*) – esta aplicação é responsável pela criação de *Traps* e notificações SNMP. Se uma *Inform* PDU for enviada, as primitivas utilizadas são do tipo *send*-PDU e *process-Response*. Caso seja enviada uma *Trap* PDU só será utilizada a primitiva *send*-PDU;

- Recetor de notificações (*Notification Receiver*) – recebe mensagens informativas e *Traps*. Esta aplicação é implementada por um *Network Management Station* (gestor). O recetor de notificações segue os mesmos procedimentos gerais do *Command Responder*. Todos os tipos de PDU's são recebidos com recurso à primitiva *process*-PDU. No processo de resposta *inform* PDU é utilizada a primitiva *return-Response*-PDU;

- Encaminhador *proxy* (*proxy Forwarder*) – facilita a transmissão de mensagens entre entidades. O encaminhador *proxy* utiliza as mesmas primitivas do módulo *Dispatcher* para enviar as mensagens SNMP e lida com os quatro tipos de mensagens compostas por PDU's distintos, produzidos pelas aplicações (*Command Genarator, Command Responder, Notification Originators, report Indicator*).

Anexo II. Estudo do protocolo NETCONF

O protocolo NETCONF (*Network Configuration Protocol*) foi desenvolvido peta IETF (*Internet Engineering Task Force*) com o propósito de unificar a forma como os equipamentos de rede são configurados, propondo padrões mais simples que as tecnologias existentes (exemplo, SNMP) e assim permitir uma gestão de rede mais eficiente [83].

Como objetivos principais, o protocolo NETCONF define um conjunto de instruções aos elementos de rede, garantindo assim, a retrocompatibilidade entre equipamentos. Estas instruções estipulam mecanismos para recolher, instalar, remover ou manipular configurações e obter estatísticas de equipamentos.

No processo de codificação das mensagens, o NETCONF recorre à linguagem XML (*eXtensible Markup Language*) [84]. O facto de o NETCONF recorrer a tecnologia XML, permite uma maior simplicidade em relação ao processo de como as informações são estruturadas. Em paralelo, o XML possibilita que os dados hierárquicos complexos sejam expressos num formato de texto, com recurso a ferramentas textuais tradicionais existentes nos sistemas.

Arquitetura do protocolo NETCONF

Antes de abordarmos a divisão da arquitetura do protocolo NETCONF, é necessário compreender o princípio de funcionamento do mesmo. Como base de funcionamento, o protocolo NETCONF utiliza o paradigma RPC (*Remote Procedure Call*), o qual estabelece um conjunto de operações. Estas operações são desempenhadas através de um cliente (agente) que codifica um pedido RPC em linguagem textual (XML), que por sua vez, envia esta informação a um servidor (gestor). No processo de resposta RPC é realizado o mesmo comportamento no sentido oposto. No fundo, este protocolo utiliza um conceito de funcionamento semelhante ao SNMP, a diferença está na arquitetura protocolar, ao qual este recorre.

Ao nível da arquitetura, esta distingue-se do SNMP, compreendendo assim o conceito de camadas. Estas camadas estão contidas no nível aplicacional do modelo TCP/IP.

O conceito arquitetónico adotado pelo protocolo NETCONF baseia-se em 4 camadas: a camada de transporte, que estabelece um meio de comunicação entre o agente e o gestor; a camada RPC, responsável pelo processamento de pedidos; a camada de operações, a qual contém um conjunto de procedimentos padrão que permitem obter dados sobre o estado e a gestão de configurações e, por fim, a camada de dados que compreende a informação de configuração e estado dos equipamentos.

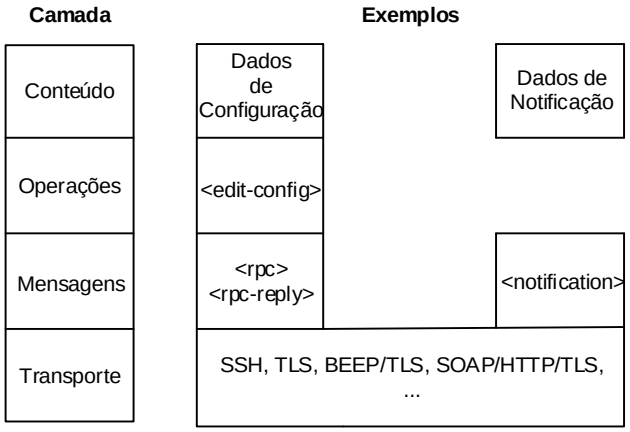


Figura 60 - Modelo arquitetónico do protocolo NETCONF

Uma particularidade do protocolo NETCONF é que este define um protocolo específico para transportar as mensagens (*Secure Shell*, SSH). No entanto, antes da RFC 6241 [83], eram utilizados outros protocolos, desde que respeitassem um conjunto de requisitos estabelecidos pela camada de transporte do protocolo NETCONF. Para atender a esses requisitos, a camada de transporte faz uso de protocolos orientados à conexão de modo a garantir a entrega dos dados, de forma sequencial e confiável. Os dados trocados entre o agente e o gestor necessitam de ser cifrados e, em paralelo, devem fornecer mecanismos de autenticação. Existem diversos protocolos que possibilitam a utilização de mecanismos de segurança, destacando-se assim, os protocolos BEEP (*Blocks Extensible Exchange Protocol*), TLS (*Transport Layer Security*) e SOAP (*Simple Object Access Protocol*) [83].

A camada RPC com base em elementos XML proporciona a execução de pedidos RPC, por parte de um agente, de modo a obter resposta do gestor. Esta utiliza a camada de transporte para realizar pedidos e atribui serviços à camada de operações.

Quando é realizada uma sessão NETCONF, é definido um elemento <*rpc*>, que tem como função atribuir um processo de execução e um atributo de identificação <*message-id*>. O gestor guarda então o pedido RPC, armazenando os valores recebidos, de forma a poder responder ao agente. No processo de resposta utiliza o elemento <*rpc-reply*>. Dentro do elemento <*rpc*>, se não existir nenhum erro na sessão RPC estabelecida, é utilizado o elemento <*ok*>, caso contrário é utilizado o elemento <*rpc-error*>.

Em relação à camada de operação, é importante salientar que existem inúmeras operações que podem ser realizadas aos equipamentos. No caso em concreto, apenas iremos abordar as operações inerentes ao processo de monitorização dos equipamentos, como por exemplo, os elementos <*copy-config*>, <*get-config*>, <*edit-config*>, <*delete-config*>, <*get*>, <*lock*>, <*unlock*> e <*kill-session*>, os quais definem métodos para consulta à base de dados, deste modo é possível obter informação de gestão do equipamento.

Anexo III. Estudo de modelos de gestão de redes

1. Gestão de redes de telecomunicações

Nas décadas de 1980 e 1990 a ITU-T (antiga CCITT, *Consultative Committee for International Telephony and Telegraphy*) desenvolveu uma arquitetura de gestão de redes de telecomunicações, bastante enquadrada com a arquitetura OSI, designada por TMN (*Telecomunication Management Network*). Esta arquitetura está definida na recomendação M.3400 [24] e é, ainda hoje, alvo de evolução e normalização, no âmbito das redes de telecomunicações de nova geração.

A TMN fornece um conjunto de operações de gestão, em particular, este sistema foi desenvolvido, de forma a possibilitar o controlo e monitorização de redes de telecomunicações. A Figura 61 ilustra o conceito de separação entre rede de gestão e a rede gerida, característica da arquitetura TMN.

O processo de gestão de redes de telecomunicações é realizado por uma rede de gestão física que é conceptualmente distinta da rede gerida.

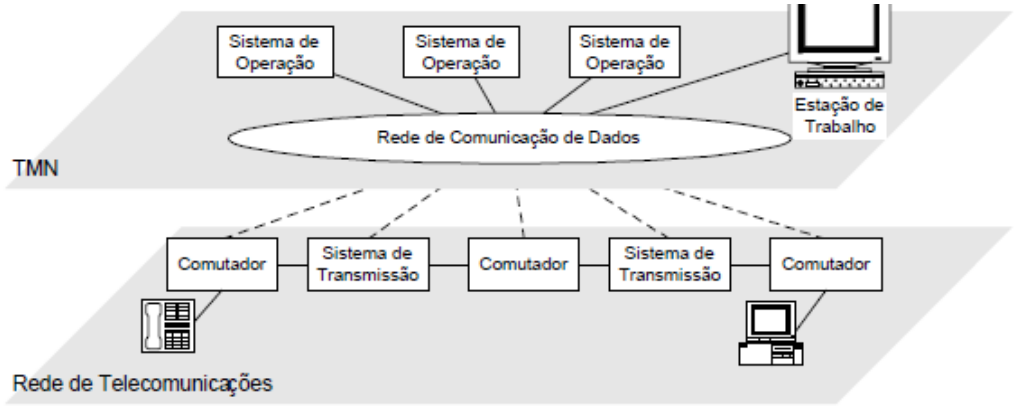


Figura 61 - Relação entre a rede de gestão TMN e a rede de telecomunicações [85]

A rede de gestão interage com pontos de comutação e transmissão da rede de telecomunicações, o que permite a comunicação entre sistemas de operações (*Operations Systems*) e o acesso a estações de trabalho (*management workstations*). A Figura 61 ilustra o conceito de separação entre rede de gestão e a rede gerida, baseada na arquitetura TMN.

O modelo de gestão TMN é segmentado em três arquiteturas distintas [85]:

- Arquitetura Funcional – especifica os blocos funcionais e os pontos de acesso aos blocos funcionais;

- Arquitetura Física – identifica as interfaces e os componentes físicos da arquitetura de gestão TMN;
- Arquitetura de Informação – descreve a aplicação dos princípios de gestão OSI numa arquitetura de gestão TMN.

Em relação à arquitetura funcional, destaca-se os seis elementos, denominados por blocos funcionais (Figura 62).

Inicialmente são realizadas operações de gestão notificações (*Operation System Funcions*, OSF) e receção, as quais, desempenham funções de gestor. No processo de transporte de informação, o mediador (*Mediation Functions*, MF) processa informação proveniente dos elementos de rede (*Network Element*, NE).

Para estabelecer a troca de blocos é utilizada a função de comunicação de dados (*Data Communication Functions*, DCF). As funções de agente são implementadas pelo bloco NEF (*Network Element Functions*), o qual está associado aos elementos de rede (NE).

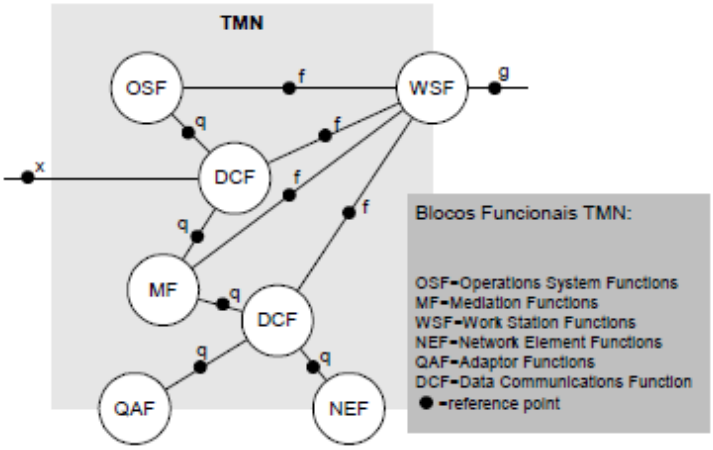


Figura 62 - Blocos funcionais e pontos de referência da arquitetura TMN

O modo de comunicação entre um bloco funcional e o utilizador é realizado com recurso ao bloco WSF (*Work Station Function*). Por fim, são utilizados adaptadores QAF (*Adaptor Functions*), de modo a possibilitar a interligação com entidades que não suportam a arquitetura de gestão TMN.

Cada bloco está associado a um componente físico e contempla uma função em concreto. Um aspeto chave sobre a arquitetura física do TMN, é as interfaces associadas. Os pontos de referência ilustrados na Figura 62 formalizam as interfaces, bem como os protocolos que constituem a implementação dos serviços do modelo OSI. As interfaces (pontos de referência) da arquitetura de gestão TMN são as seguintes [85]:

- F – Implementadas entre TMN e as estacoes de trabalho;
- Q – Existentes entre sistemas de gestão TMN;
- X – Localizadas entre componentes de diferentes redes TMN;
- Qx – Presentes entre elementos de rede e sistemas de mediação;
- G – Pertencentes entre as estações de trabalho e o utilizador.

2. Gestão de redes baseada em *web*

A tecnologia *web* propõe uma representação de informação, assente em plataformas e atividades de gestão com diversas vantagens, das quais as principais são a grande divulgação, aceitação e disponibilidade das tecnologias base, a independência em relação aos sistemas operativos e, por fim, a capacidade para a integração de soluções de gestão de redes.

A referência de gestão baseada em *Web* é nomeada *Web Based Enterprise Management* (WBEM), e foi desenvolvido pela DMTF (*Distributed Management Task Force*). A tecnologia WBEM é amplamente utilizada nos dias de hoje e tem uma larga aceitação por parte dos fabricantes de equipamentos/*software*. O modelo WBEM recorre a três princípios fundamentais:

- Modelo de informação baseado no *Common Information Model* (CIM);
- Codificação de objetos com recurso à linguagem XML e utilização de arquitetura CORBA (*Common Object Request Broker Architecture*);
- Transporte da informação de gestão com auxílio do protocolo HTTP.

CIM - *Common Information Model*

O modelo de informação comum (CIM) foi desenvolvido pela DMTF. Ao contrário do modelo MIB, que utiliza um conjunto de objetos para definir um processo de gestão, o modelo CIM, permite utilizar técnicas de programação orientada a objetos. O conceito do modelo CIM traduz-se essencialmente num esquema que descreve uma estrutura de gestão de informação. Os esquemas são representados visualmente com uma linguagem de modelação unificada (*Unified Modelling Language*, UML) [86], e utilizam uma linguagem de programação MOF (*Managed Object Format*). No que respeita à estruturação do CIM, este divide-se em três camadas distintas [87]:

- Modelo central (*Core model*) – define um conjunto de classes básicas para as diferentes áreas de gestão;
- Modelo comum (*Common model*) – cria um conjunto de classes comuns, para diferentes áreas de gestão, como sistemas, aplicações, redes e dispositivos. O conjunto de classes define as informações de gestão que podem ser estendidas para tecnologias específicas;
- Esquemas de extensão (*Extension schemes*) – identifica extensões específicas da tecnologia CIM, por exemplo, uma extensão para um sistema operativo (Windows, UNIX, entre outros).

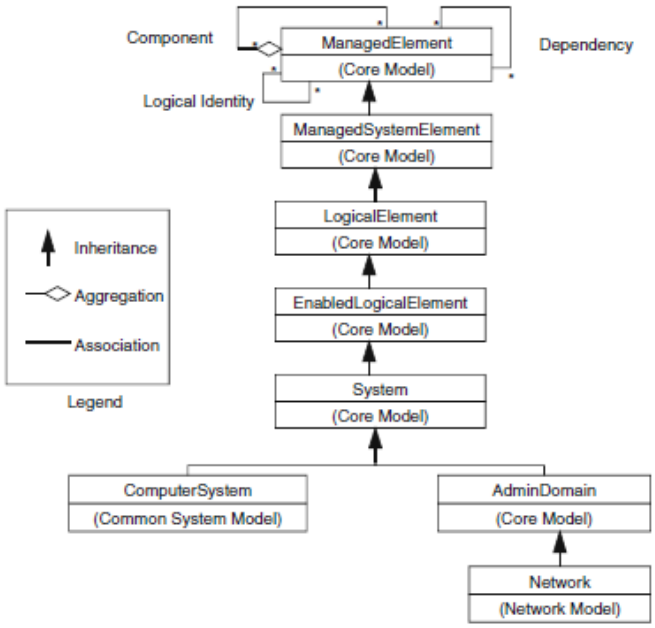


Figura 63 - Hierarquia de classes principais do modelo CIM

O modelo CIM é composto por diversas classes conforme ilustra a Figura 63, no entanto

apenas serão descritas as principais. A classe *Managed-Element* define a raiz no modelo CIM. Todas as restantes classes derivaram desta classe. Um elemento gestão pode ter vários relacionamentos com outros elementos geridos. O elemento *Managed-Element* é uma classe abstrata com atributos únicos (nome e uma descrição).

Os elementos *Managed-System-Elements* podem ser definidos por entidades físicas (*Physical-Element*), por exemplo, interfaces de equipamentos, servidores, ou entidades lógicas (*Logical-Element*) como por exemplo protocolos de transporte TCP/UDP.

CORBA - *Common Object Request Broker Architecture*

A arquitetura CORBA (*Common Object Request Broker Architecture*)[88] foi desenvolvida para dar resposta à interoperabilidade entre diferentes produtos, nomeadamente ao nível do *software* e *hardware*, possibilitando assim, o acesso à informação de uma forma transparente, sem existir a necessidade de conhecer a sua plataforma ou o protocolo de transporte. A composição da arquitetura CORBA é definida por objetos servidores, objetos clientes e um intermediário (*Object Request Broker*, ORB), o qual, estabelece uma relação entre os objetos e interseta as invocações, assumindo também, a responsabilidade no processo de descoberta do objeto pretendido, de forma a devolver resultados. O ORB também pode ser visto como uma base de apoio a operações fundamentais para gestão dos objetos distribuídos [89].

No modelo de funcionamento do CORBA são trocadas mensagens sob a forma de invocações de métodos relacionados com diferentes objetos. A descrição de cada mensagem é realizada com base na construção de uma interface que especifica o tipo de pedidos que o objeto pode intersetar. Independentemente da linguagem adotada, uma interface é definida por uma linguagem própria (*Interface Definition Language*, IDL). Isto possibilita a comunicação entre componentes com diferentes linguagens de programação [88].

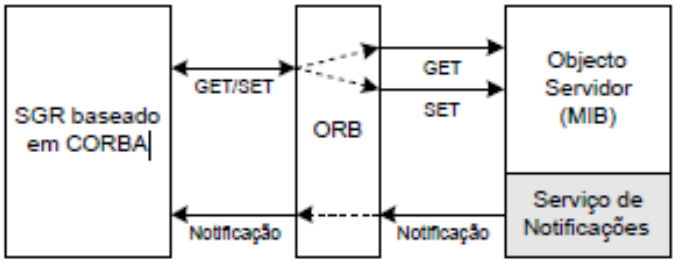


Figura 64 - Comunicação entre gestor e agente baseados em CORBA

Com recurso à Figura 64, podemos verificar um cenário de uma gestão baseada em CORBA. Tradicionalmente o CORBA utiliza no seu processo de comunicação o mesmo

modelo genérico da arquitetura SNMP. O comportamento dos agentes é determinado pelas interfaces IDL, as quais são invocadas pelo gestor (Sistema de Gestão de Redes, SGR). Como intermediário deste processo é utilizado o ORB, que recorre a um protocolo geral de interoperabilidade (*General Inter-ORB Protocol*, GIOP) e assegura assim a correspondência entre o GIOP e o protocolo de transporte (TCP ou IPX, *Internetwork Packet Exchange*, por exemplo).

Ao nível das operações, são realizadas consultas (*GET*) e pedidos de modificação (*SET*) do estado agente. O modelo de informação utilizado por cada agente é específico e fornece uma base de apoio às solicitações realizadas pelo gestor. O agente é baseado num sistema CORBA (servidor) que envia notificações (*TRAP*), utilizando para o efeito um serviço de notificações.

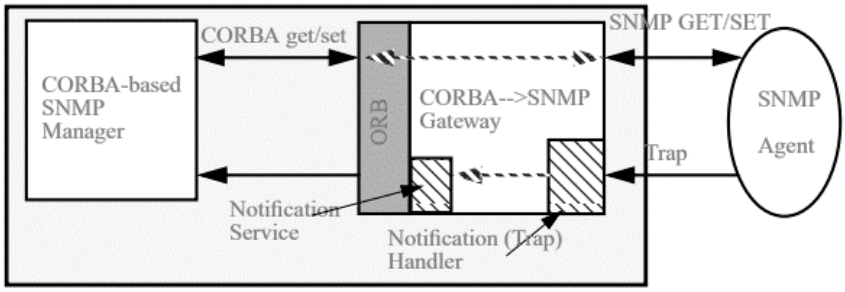


Figura 65 - Integração do CORBA baseado em gestor e agente SNMP [90]

O protocolo SNMP pode ser integrado com o CORBA, através do modelo de informação, fazendo a correspondência entre as MIB's e as interfaces IDL [90]. A interface IDL converte então os pedidos CORBA para PDU's SNMP e vice-versa (Figura 65).

3. Gestão de redes baseada em políticas

A gestão de rede baseada em políticas (*Policy-based network management*, PBNM) assenta num princípio que permite gerir a configuração e o comportamento de uma ou mais entidades, com base nas necessidades e políticas de negócio. Deste modo, este paradigma define dois níveis de abstração [91]:

- Nível superior, o qual é autónomo das tecnologias ou equipamentos, e estabelece regras (políticas) que instituem o comportamento das redes e sistemas;
- Nível de execução das políticas, onde é feito o mapeamento entre as políticas do nível superior e as tecnologias específicas do equipamento, redes e sistemas.

Na componente prática, o processo de gestão de rede baseada em políticas é consolidado pelo protocolo *Common Open Policy Service* (COPS) [92], o qual é responsável pela

comunicação entre os dois níveis de abstração (nível superior e nível de execução das politicas), conforme é ilustrado na Figura 66.

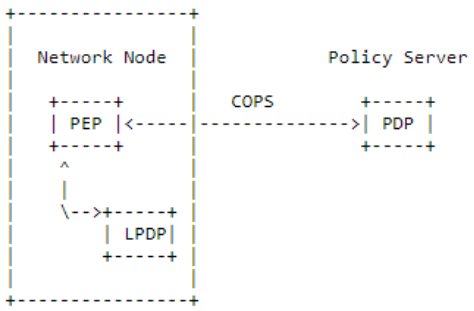


Figura 66 - Modelo COPS

Existem três entidades essenciais que operam no modelo COPS [93]:

- *Policy Decision Point* (PDP) – implementa um ponto central de gestão, encarregue de tomar decisões de política de gestão, com base num repositório de políticas e subsistema de interação com operadores humanos;
- *Policy Enforcement Point* (PEP) – refere-se a uma entidade que aplica as políticas de gestão, mapeando-as em regras específicas do equipamento;
- *Local Policy Decision Point* (LPDP) – comporta decisões locais, quando não existe um PDP, isto acontece no modelo COPS-PR (COPS *Usage for Provisioning*).

O modelo COPS trata-se de um protocolo em modo pedido-resposta, que possibilita que os PEP interroguem um PDP. Para evitar a consulta constante dos PDP por parte dos PEP, foi desenvolvido uma extensão ao protocolo COPS, nomeadamente, o protocolo COPS-PR [94].

Isto possibilitou que os PDP passassem a fornecer a informação aos PEP, de uma só vez. Os PEP passaram a tomar decisões com base em políticas previamente carregadas pelos PDP e armazenadas localmente nos LPDP.

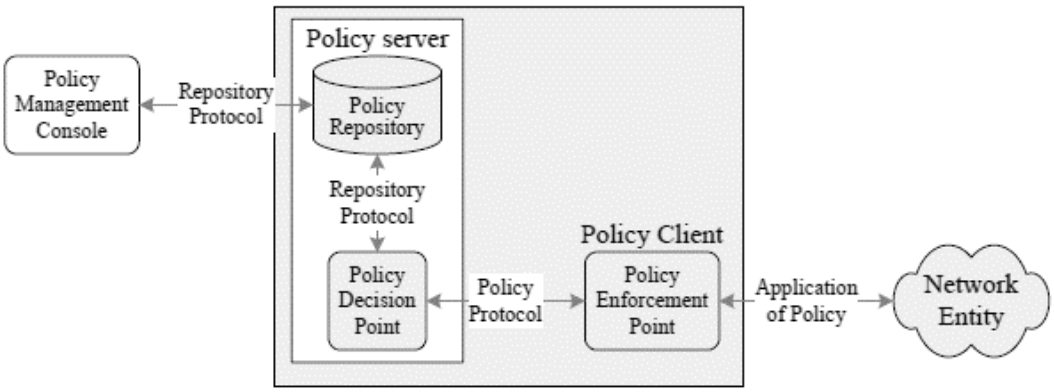


Figura 67 - Modelo lógico de arquitetura PBNM

Na Figura 67 é demonstrado um modelo geral da arquitetura definida pelo PBNM. A entidade *Policy Management Console* (PMC) faz de interface entre o utilizador e o repositório de políticas, e é responsável por implementar regras de forma a monitorizar o ambiente de gestão. O PDP mediante as regras anteriormente definidas pelo PMC determina as decisões a serem tomadas. Com auxílio do protocolo COPS é realizada comunicação entre o PDP e o PEP. Por fim, o PEP através de políticas de aplicação, controla as políticas de gestão definindo as regras específicas do equipamento.

Anexo IV. Configuração do protocolo OSPF IPv4 no ASR II

Este anexo retrata as principais configurações da rede de endereçamento IPv4 da U. Porto. A inclusão da interface BVI (*Bridge Virtual interface*) possibilita que a VRF (IPv4_publico) possa alcançar todos os prefixos de rede IP públicos, tendo por base as subinterfaces físicas associadas a outra instância OSPF.

```
interface BVI1
 description ***** netUP IPv4 Routing *****
 vrf IPv4_publico
 ipv4 address 191.40.3.2 255.255.255.240
```

Figura 68 - Configuração da interface BVI para o processo OSPF-1 IPv4

```
router ospf 2
 nsr
 vrf IPv4_publico
  log adjacency changes detail
  router-id 191.40.3.2
  default-information originate metric-type 1
  redistribute connected
  redistribute static
  area 0.0.0.0
   interface BVI1
    authentication message-digest
    message-digest-key 1 md5 encrypted
    dead-interval 3
    hello-interval 1
   !
  !
 !
!
vrf netupipv4
  address-family ipv4 unicast
   <prefix x.x.x.x/x> <x.x.x.x next-hop> description <texto>
```

Figura 69 - Configuração do processo OSPF-2 IPv4

Com a inclusão do *router ospf* <ID>, é determinado o número do processo. Concretamente as rotas são redistribuídas de duas formas, isto é, as *redes directly connected* e as *static routes* são passadas aos *neighbors* do processo OSPF

Anexo V. Configuração do protocolo BGP no ASR II

As próximas figuras, visam demonstar as principais configurações do protocolo BGP existente nos *routers* ASR que se interligam à rede da RCTS.

```
vrf IPv4_publico
  rd 65104:4
  bgp router-id 193.136.56.252
  address-family ipv4 unicast
   network <redes sumarizadas>
   redistribute ospf 2 route-policy BGP_Filter_OUT
  !
  neighbor 80.10.1.1
   remote-as 1900
   timers 20 60
   update-source BVI110
   address-family ipv4 unicast
    weight 200
    route-policy BGP_Filter_IN in
    route-policy BGP_Filter_OUT out
    next-hop-self
    soft-reconfiguration inbound
   !
  !
  neighbor 193.136.56.253
   remote-as 65100
   timers 20 60
   update-source BVI10
   address-family ipv4 unicast
    weight 150
    route-policy Accept-All in
    route-policy Accept-All out
    next-hop-self
    soft-reconfiguration inbound
   !
  !
!
```

Figura 70 - Configuração do Protocolo BGP

Embora todos os ASRs pertençam ao grupo de AS privado (*route-distinguisher*, <AS:ID>, 65100:4), apenas o ASRII e o ASRIII se interligam à rede da RCTS, estabelecendo sessões EBGP com o *remote-as* público (1900).

Cada *neighbor* está inserido num único processo BGP, recorrendo ao endereço lógico. As interfaces lógicas de layer 3 (BVI) são configuradas com o IP local às redes de interligação, sendo este definido no *update-source*.

```
interface BVI110
 description **** Ligacao RCTS IPv4 ****
 vrf IPv4_publico
 ipv4 address 80.100.1.2 255.255.255.252
!
interface BVI10
 description **** Ligacao ASRII-to-ASRIII IPv4 ****
 vrf IPv4_publico
ipv4 address 193.136.56.252 255.255.255.248
!
```

Figura 71 - Configuração de interfaces BVI no ASRII

Tipicamente, um protocolo de encaminhamento exterior assenta na troca de informação relativa aos caminhos (*paths*).

```
route-policy BGP_Filter_IN
  if destination in default-only then
    pass
  else
    drop
  endif
end-policy
!
route-policy BGP_Filter_OUT
  if destination in BGP_IPv4_uporto_announce then
    pass
  else
    drop
  endif
end-policy
!
prefix-set default-only
  0.0.0.0/0
end-set
!
route-policy Accept-All
  pass
end-policy
```

Figura 72 - Configuração das políticas de rotas BGP

A tomada de decisões do protocolo BGP é baseada em “políticas” de encaminhamento, configuradas em *route-policys*. Deste modo, cada filtro determina aquilo que é recebido (IN) ou enviado (OUT).

Anexo VI. Configuração do protocolo MPLS no ASRII

Toda a configuração de rede da U. Porto tem como base o MPLS. Isto origina a necessidade de um IGP como referido anteriormente, sendo adotado o protocolo OSPF.

O processo OSPF-1 existente não recorre a nenhuma VRF de modo a viabilizar apenas a criação de adjacências entre os *neighbors*, a partilha das redes de interligação e os endereços *loopback* de identificação dos ASRs no MPLS.

```
interface Loopback1
 ipv4 address 172.18.0.2 255.255.255.255
 ipv6 enable
!
```

Figura 73 - Configuração de interface *loopback*

```
router ospf 1
 nsr
 log adjacency changes detail
 router-id 172.18.0.2
 default-information originate metric-type 1
 area 0.0.0.0
  mpls ldp sync
  interface Loopback1
  !
  interface TenGigE0/0/0/0.1
  !
  interface TenGigE0/0/0/0.3
  !
 !
```

Figura 74 - Configuração do processo OSPF-1 MPLS

Em parelelo, esta instância OSPF-1 auxilia o LDP (*mpls ldp sync*), onde a configuração define a identificação das interfaces locais, bem como a interface *loopback* do nó em causa. Estas interfaces irão realizar a troca de etiquetas que se pretendem distribuir dinamicamente e, ao mesmo tempo, irão partilhar todos os prefixos de rede com os endereços *loopback* (*router id*) associados à área 0.0.0.0.

```
 mpls ldp
  router-id 172.18.0.2
  nsr
  graceful-restart
  session protection for LDP-PEERS
  explicit-null
  igp sync delay 10
  !
  interface TenGigE0/0/0/0.1
  !
  interface TenGigE0/0/0/0.3
  !
 !
 mpls ip-ttl-propagate disable
 !
 ipv4 access-list LDP-PEERS
  10 permit ipv4 host 172.18.0.1 any
  20 permit ipv4 host 172.18.0.2 any
  30 permit ipv4 host 172.18.0.3 any
  30 permit ipv4 host 172.18.0.4 any
```

Figura 75 - Configuração do protocolo MPLS/LDP

A configuração do MPLS referente ao LDP, tem como particularidade descrever como o mecanismo das etiquetas irá ser realizado para a normal criação e identificação dos LSPs, assim como, para a posterior sinalização dos *pseudowires* (serviços VPLS).

A inclusão do método *explicit-null* pressupõe a utilização de uma etiqueta de valor reservado (zero), obrigando a que seja executada a operação de *penultimate hop popping*. Deste modo, o penúltimo nó informa o último (numa comunicação entre três nós) que deve retirar a etiqueta do cabeçalho MPLS e o pacote é entregue como base no cabeçalho IP.

```
 router bgp 65100
  nsr
  bgp router-id 172.18.0.2
  neighbor-group mbgp-session-PEERs
   remote-as 65100
   password encrypted <secret>
   update-source Loopback1
 !
!
```

Figura 76 - Configuração do processo M-BGP 1/2

```
neighbor 172.18.0.1
  use neighbor-group mbgp-session-PEERs
  description Pólo 1 - Reitoria
 !
 neighbor 172.18.0.3
  use neighbor-group mbgp-session-PEERs
  description Pólo 3 - FCUP
 !
 neighbor 172.18.0.4
  use neighbor-group mbgp-session-PEERs
  description Pólo 4 - FDUP
 !
```

Figura 77 - Configuração do processo M-BGP 2/2

O mecanismo de descoberta dos *peers* MPLS recorre ao processo BGP previamentre configurado. Embora o processo BGP seja só um, este protocolo associado ao MPLS ganha o nome M-BGP e possibilita o conceito *Auto Discovery*. Destaca-se então, como uma mais-valia para a criação de múltiplos serviços, não obrigando à defenição dos nós de transporte por cada serviço VPLS instanciado (consultar Anexo VII).

Anexo VII. Configuração de serviços VPLS no ASR I

```
interface Bundle-Ether1.100 l2transport
description ***gestão KVM pólo 1***
encapsulation dot1q 100
!
interface Bundle-Ether4.100 l2transport
description ***gestão KVM pólo 1***
encapsulation dot1q 100
!
interface Bundle-Ether4.200 l2transport
description ***gestão KVM pólo 2***
encapsulation dot1q 200
!
interface Bundle-Ether4.300 l2transport
description ***gestão KVM pólo 3***
encapsulation dot1q 300
!
interface Bundle-Ether5.50 l2transport
description ***FW gestão failover***
encapsulation dot1q 50
!
```

Figura 78 - Configuração dos *attachment-circuit*

```
l2vpn
ignore-mtu-mismatch-ad
bridge-group netUP_Management
bridge-domain gestão-KVM-pólo-1
interface Bundle-Ether1.100
!
interface Bundle-Ether4.100
!
   vfi gestão-KVM-pólo-1
    vpn-id 100
    autodiscovery bgp
     rd 65104:100
     route-target 65104:100
     signaling-protocol ldp
     !
    !
   !
```

Figura 79 - Configuração dos serviços – VPLS 1/2

```
bridge-domain gestão-KVM-pólo-2
interface Bundle-Ether4.200
!
vfi gestão-KVM-pólo-2
  vpn-id 200
   autodiscovery bgp
    rd 65104:200
    route-target 65104:200
    signaling-protocol ldp
     !
    !
   !
bridge-domain gestão-KVM-pólo-3
interface Bundle-Ether4.300
!
vfi gestão-KVM-pólo-3
  vpn-id 300
   autodiscovery bgp
    rd 65104:300
    route-target 65104:300
    signaling-protocol ldp
     !
    !
   !
bridge-domain fw-gestao-failover
interface Bundle-Ether5.50
!
vfi fw-gestao-failover
  vpn-id 50
   autodiscovery bgp
    rd 65104:50
    route-target 65104:50
    signaling-protocol ldp
     !
    !
   !
```

Figura 80 - Configuração dos serviços – VPLS 2/2

Anexo VIII. Configuração de *Virtual Private Networks* em *firewalls* ASA

Tendo em conta o exemplo de comunicação retratado na Figura 27, as configurações que se seguem pretendem enumerar os tipos de VPNs presentes na rede de comunicação de dados da U. Porto. O cenário adotado envolve duas *firewalls* Cisco ASA (*Adaptive Security Appliance*) com versões de *hardware* (modelo, 5545 e 5540) e de *software* distintas (*firmware*, 9.4 e 8.2)

1. Túneis IPSec *site-to-site*

As seguintes figuras demonstram a configuração de um túnel IPsec, entre a *firewall* de gestão (versão 9.4) e a *firewall* do Pólo 1 (versão 8.2). Pretende-se representar como a rede de gestão consegue ter acesso a uma ou mais redes do pólo 1, através de um túnel lógico estabelecido entre as interfaces *outside* das *firewalls*.

```
  #FW-gestao:
  !
  #access-list inside:
  access-list gestao extended permit ip host object-group NET-gestao object-group NET-pólo1
  !
  #access-list tunel
  access-list ipsec_to_p1_fw1 extended permit ip object-group NET-gestao object-group NET-pólo1
  !
  #NAT
  nat (up_net,outside) source static NET-gestao NET-gestao destination static NET-pólo1 NET-pólo1
  !
  #-----------------------------------------------------------------------------------------#
  #FW-pólo1
  !
  #access-list tunel
  access-list ipsec_to_fw_gestao extended permit ip object-group NET-gestao object-group NET-pólo1
  !
  #NAT zero
  nat (service_pólo1) 0 access-list service_pólo1_nat0_outbound
  access-list service_pólo1_nat0_outbound extended permit ip object-group NET-pólo1 object-group
NET-gestao
```

Figura 81 - Configuração de *access-lists* e NATs

Neste caso em concreto, é pretendido que a rede de origem (*object-group* NET-gestao) da *firewall* de gestão, tenha acesso a todas as portas (*permit* IP) da rede destino (*object-group* NET-pólo1) presente na *firewall* do pólo 1.

No caso da *firewall* de gestão, a versão de *software* é superior à 8.4, o que possibilita que diferentes interfaces com *security levels* distintos, consigam comunicar-se sem o recurso a mecanismos de NAT zero. Por sua vez, o NAT estático substitui o NAT zero, sendo de uso obrigatório para traduzir endereços IP associados a um túnel VPN.

Devido à *firewall* do pólo 1 ter uma versão inferior à 8.4, necessita do modelo de NAT "zero", sendo que o túnel configurado recorre à interface *outside*, a qual, tem um *security level* diferente da interface *inside* onde reside a rede de destino a atingir.

```
#FW-gestao
crypto ikev1 policy 10
 authentication pre-share **password**
 encryption 3des
 hash md5
 group 2
 lifetime 86400
!
crypto ikev1 enable outside
crypto isakmp identity hostname
!
#-------------------------------------------------------------------------------------------#
#FW-pólo1
crypto isakmp policy 10
 authentication pre-share **password**
 encryption 3des
 hash md5
 group 2
 lifetime 86400
!
crypto isakmp enable outside
crypto isakmp identity hostname
```

Figura 82 - Configuração do protocolo IKEVv1/ISAKMP

O protocolo ISAKMP (*Internet Security Association and Key Management Protocol*) define uma estrutura de autenticação (*authentication pre-share key*) e troca de chaves encriptadas < {aes | aes-192 | aes-256 | 3des | des} > para os *peers* estabelecerem um canal de comunicação IPsec.

Embora o protocolo ISAKMP seja compatível com o IKE (*Internet Key Exchange*), a diferença cinge-se no tipo de configuração que a CISCO propõe. As versões anteriores à 8.4, não possibilitam a utilização do IKE. Isto é, a política de criptografia utilizada na *firewall* do pólo1 recorre ao protocolo ISAKMP, sendo que esta estrutura protocolar dá suporte nativo ao IKE e, por isso, existe a retrocompatibilidade entre estes dois mecanismos criptográficos.

A função de *hash* < {sha | md5} > é o algoritmo que mapeia "*strings*" para números inteiros, ajudando a manter a integridade dos dados e, consequentemente, fornece uma camada extra no que concerne aos métodos de segurança.

O *diffie hellman* < group {1 | 2 | 5 | 7} > assegura a confidencialidade com base no processo de compartilhamento da chave criptográfica simétrica. Cada ID de um grupo tem de ser configurado de igual modo nas *firewalls* que partilham o mesmo túnel IPsec. Quanto maior for o ID, maior será o número de *bits* que a cifra disponibiliza, resultando assim, num método mais seguro de comunicação. É de salientar que quanto maior for o valor do ID *group*, maior será o aumento de latência nos equipamentos derivado ao aumento significativo de recursos de CPU.

```
#FW-gestao
tunnel-group 193.136.25.10 type ipsec-l2l
tunnel-group 193.136.25.10 ipsec-attributes
 ikev1 pre-shared-key **secret**
!
#FW-pólo1
tunnel-group 193.136.25.42 type ipsec-l2l
tunnel-group 193.136.25.42 ipsec-attributes
 pre-shared-key **secret**
```

Figura 83 - Configuração do túnel IPsec *site-to-site*

O endereço IP de cada interface *outside* das *firewalls* deve ser atribuído a um *tunnel-group* para que o canal lógico possa ser construído. Concretamente, os parâmetros < {ipsec-l2l | remote-access} > do tipo de túnel irão determinar o modo de operação, podendo este ser *site-to-site* ou *remote-access*.

2. Túneis IPSec *remote-to-access*

O tipo de acesso remoto por via do utilizador pode ser efetuado com recurso a um configurador automático, ou através da configuração "manual" de VPNs num dado sistema computacional. Concretamente, a configuração que se segue, pretende descrever as parametrizações necessárias para o estabelecimento de um túnel IPSec entre a *firewall* de gestão e o sistema terminal do administrador de rede. É importante referir, que não será descrita a metodologia para *web* VPN, a qual propõe configurações auxiliares na *firewall*, de forma a possibilitar a utilização de um instalador automático de VPNs.

As configurações deste tipo de túnel são bastante similares às previamente abordadas nos túneis IPsec *site-to-site*. Uma das principais diferenças recai na necessidade da configuração adicional de um *range* de endereços IP (VPN *pool*), com o objetivo de um dado utilizador ficar com um endereço IP atribuído dinamicamente pela *firewall*.

```
  ip local pool VPNPOOL 192.168.20.1-192.168.20.254
  !
  # configurção split tunneling (OPTIONAL)
  access-list    splittunnel    standard    permit    192.168.20.0    255.255.255.0    object-group
gestao_equipamentos
```

Figura 84 - Configuração de *poo*l de endereços VPN *remote-to-access*

Para permitir o acesso da rede 192.168.20.0/24 (VPN *pool*) às redes das interfaces *inside* da *firewall*, deve-se configurar o mecanismo de *split-tunnel*. Do ponto de vista teórico o modelo de *split-tunnel* baseia-se no acesso às redes que estiverem permitidas numa mera *access-list* convencional interligada a uma *group-policy*. Isto pressupõe que todos os prefixos de rede que não tiverem na tabela de rotas do sistema computacional ligado à VPN, terão como *defaut gateway* o endereço IP do dispositivo de rede local (LAN).

Por sua vez, o modelo de *tunnelall*, preconiza um mecanismo contrário ao do *split-tunnel*. Todos os prefixos de rede que não estão presentes na tabela de rotas do sistema terminal, são alcançáveis através da *firewall*. Assim, o utilizador tem instanciadas no seu sistema, duas rotas *defualt*, sendo que uma delas é a preferível (métrica menor) e tem como endereço de *next-hop* a interface associada ao túnel IPSec (*outside*). Como boa prática, a administração e gestão de equipamentos com recurso a uma VPN do tipo *remote-to-access*, deve utilizar o modo *tunnelall*, garantindo que todo o tráfego de destino sai sempre por um único ponto.

Anexo IX. Modelos de gestão de armazenamento PostgreSQL *versus* MySQL

Tabela 35 - Principais diferenças entre arquiteturas PostgreSQL e MySQL [66]

Feature	***PostgreSQL***	***MySQL***
Open Source	*Completely Open source*	*Open source, but owned by Oracle and offers commercial versions*
ACID Compliance	*Complete ACID Compliance*	*Some versions are compliant*
SQL Compliance	*Almost fully compliant*	*Some versions are compliant*
Concurrency Support	*MVCC implementation supports multiple requests without read locks*	*Support in some versions*
Security	*Secure from ground up with SSL support*	*SSL support in some versions*
NoSQL/JSON Support	*Multiple supported features*	*JSON data support only*
Access Methods	*Supports all standards*	*Supports all standards*
Replication	*Multiple replication technologies available:* • *Single master to one standby* • *Single master to multiple standbys* • *Hot Standby/Streaming Replication* • *Bi-Directional replication* • *Logical log streaming replication*	*Standard master-standby replication:* • *Single master to one standby* • *Single master to multiple standbys* • *Single master to one standby to one or more standbys* • *Circular replication (A to B to C and back to A)* • *Master to master*
Materialized Views	*Supported*	*Not supported*
Temporary Tables	*Supported*	*Supported*
GeoSpatial Data	*Supported*	*Supported*
Programming Languages	*Supported*	*Not supported*

Tabela 36 - Análise de desempenho entre arquiteturas PostegreSQL e MySQl [67]

PostgreSQL	***MySQL***
PostgreSQL is widely used in large systems where read and write speeds are crucial and data needs to validated. In addition, it supports a variety of performance optimizations that are available only in commercial solutions such as *Geospatial data support, concurrency without read locks, and so on (e.g. Oracle, SQL Server).* *Overall, PostgreSQL performance is utilized best in systems requiring execution of complex queries.* *PostgreSQL performs well in OLTP (Online Transaction Processing)/OLAP (Online Analytical Processing) systems when read/write speeds are required and extensive data analysis is needed.* *PostgreSQL also works well with Business Intelligence applications but is better suited for Data Warehousing and data analysis applications that require fast read/write speeds.*	*MySQL is a widely chosen for web based projects that need a database simply for straightforward data transactions. It is common, though, for MySQL to underperform when strained by a heavy loads or when attempting to complete complex queries.* *MySQL performs well in OLAP/OLTP systems when only read speeds are required.* *MySQL + InnoDB provides very good read/write speeds for OLTP scenarios. Overall, MySQL performs well with high concurrency scenarios.* *MySQL is reliable and works well with Business Intelligence applications, as business intelligence applications are typically read-heavy.*

Anexo X. Estudo comparativo entre soluções *open-source* MySQL

Tabela 37 - Funcionalidades entre MySQL, Percona e MariaDB 1/4 [69]

Feature	***MySQL***	***Percona***	***MariaDB***
Protocols	*MySQL protocol over port 3306, X Protocol over port 33060*	*MySQL protocol over port 3306, X Protocol over port 33060*	*MySQL protocol, MariaDB Server extensions*
Community – Source Code	*Open Source*	*Open Source*	*Open Source*
Community – Development	*Open Source, contributions via signing the Oracle Contributor Agreement (OCA)*	*Open Source*	*Open Source, contributions via the new BSD (Berkeley Software Distribution) license or signing the MariaDB Contributor Agreement (MCA)*
Community – Collaboration	*Mailing list, forums, bugs system*	*Mailing list, forums, bugs system (Jira, Launchpad)*	*Mailing list, bugs system (Jira), IRC (Internet Relay Chat) channel*
Core – Replication	*MySQL replication with GTID*	*MySQL replication with GTID*	*MariaDB Server replication, with own GTID, compatible only if MariaDB Server is a slave to MySQL, not vice versa*
Core – Routing	*MySQL Router (GPLv2)*	*proxySQL (GPLv3)*	*MariaDB MaxScale (Business Source License)*
Core – Partitioning	*Standard*	*Standard*	*Standard, with extra engines like SPIDER/CONNECT that offer varying levels of support*
Tool – Editing	*MySQL Workbench for Microsoft Windows, macOS, and Linux*	*MySQL Workbench for Microsoft Windows, macOS, and Linux*	*Webyog's SQLYog for Microsoft Windows (MySQL Workbench notes an incompatible server)*
Tool – Monitoring	*MySQL Enterprise Monitor*	*Percona Monitoring & Management (PMM) (100% open source)*	*Webyog's Monyog*

Tabela 38 - Funcionalidades entre MySQL, Percona e MariaDB 2/4 [69]

Feature	***MySQL***	***Percona***	***MariaDB***
Scalability – clustering	*MySQL Group Replication*	*MySQL Group Replication, Percona XtraDB cluster (based on a further engineered Galera cluster)*	*MariaDB Enterprise cluster (based on Galera cluster)*
Security – Encryption	*Tablespace data-at-rest encryption. Amazon KMS (Key Management Service), Oracle Vault Enterprise Edition*	*Tablespace data-at-rest encryption with Keyring Vault plugin*	*Tablespace and table data-at-rest encryption. Amazon KMS, binlog/redo/tmp file with Aria tablespace encryption*
Security – Data Masking	*proxySQL data masking*	*proxySQL data masking*	*MariaDB MaxScale data masking*
Security – Firewall	*MySQL Enterprise Firewall*	*proxySQL Firewall*	*MariaDB MaxScale Firewall*
Security – Auditing	*MySQL Enterprise Audit Plugin*	*Percona Audit Plugin (OSS)*	*MariaDB Audit Plugin (OSS)*
Analytics	*No*	*ClickHouse*	*MariaDB ColumnStore*
SQL – Common Table Expressions	*In-development for MySQL 8.0 (now a release candidate)*	*In-development for MySQL 8.0 (now a release candidate)*	*Present in MariaDB Server 10.2*
SQL – Window Functions	*In-development for MySQL 8.0 (now a release candidate)*	*In-development for MySQL 8.0 (now a release candidate)*	*Present in MariaDB Server 10.2*
Temporal – Log-based rollback	*No*	*No*	*In development for MariaDB Server 10.3*
Temporal – system versioned tables	*No*	*No*	*In development for MariaDB Server 10.3*
JSON	*JSON Data type, 21 functions*	*JSON Data type, 21 functions*	*No JSON Data Type, 26 functions*
Official client connectors	C (libmysqlclient), Java, ODBC, .NET, Node.js, Python, C++, mysqlnd for PHP	C (libmysqlclient), Java, ODBC, .NET, Node.js, Python, C++, mysqlnd for PHP	C (libmariadbclient), Java, ODBC

Tabela 39 - Funcionalidades entre MySQL, Percona e MariaDB 3/4 [69]

Feature	***MySQL***	***Percona***	***MariaDB***
Usability – CJK Language support	*Gb18030, ngram & MeCab for InnoDB full-text search*	*Gb18030, ngram & MeCab for InnoDB full-text search*	*No*
Monitoring – PERFORMANCE _SCHEMA	*Thorough instrumentation in 5.7, sys schema included*	*Thorough instrumentation in 5.7, sys schema included*	*Instrumentation from MySQL 5.6, sys schema not included*
Security – Password authentication	*sha256_password (with caching_sha2_password in 8.0)*	*sha256_password (with caching_sha2_password in 8.0)*	*ed25519 (incompatible with sha256_password)*
Security – Secure out of the box	*validate_password on by default, to choose a strong password at the start*	*validate_password on by default, to choose a strong password at the start*	*No*
Usability – Syntax differences	*EXPLAIN FOR CONNECTION <thread_id>*	*EXPLAIN FOR CONNECTION <thread_id>*	*SHOW EXPLAIN FOR <thread_id>*
Optimiser – Optimiser Tracing	*Yes*	*Yes*	*No*
Optimiser – Optimiser Hints	*Yes*	*Yes*	*No*
DBA – Super readonly mode	*Yes*	*Yes*	*No*
Security – Password expiry	*Yes*	*Yes*	*No*
Security – Password last changed? Password lifetime?	*Yes*	*Yes*	*No*
Security – VALIDATE_PASSWORD _STRENGTH()	*Yes*	*Yes*	*No*
Security – ACCOUNT LOCK/UNLOCK	*Yes*	*Yes*	*No*
Usability – Query Rewriting	*Yes*	*Yes*	*No*
GIS – GeoJSON & GeoHash functionality	*Yes*	*Yes*	*Incomplete*

Tabela 40 - Funcionalidades entre MySQL, Percona e MariaDB 4/4 [69]

Feature	***MySQL***	***Percona***	***MariaDB***
Security – mysql_ssl_rsa_setup	*Yes*	*Yes*	*No (setup SSL connections manually)*
MySQL Utilities	*Yes*	*Yes*	*No*
Backup locks	*No (in development for 8.0)*	*Yes*	*No*
Usability – InnoDB memcached interface	*Yes*	*Yes*	*No*

Anexo XI. Estudo comparativo de ferramentas de *backup* do MySQL

Tabela 41 - Funcionalidade de *backups* Percona SSTv2 vs MySQL [95] 1/3

Recursos	**Percona XtraBackup**	***Backup* do MySQL Enterprise**
Licença	GPL (*General Public License*)	Proprietário
Preço	Livre	Incluído na assinatura por US $ 5000 por servidor
Formatos de streaming e criptografia	Código aberto	Proprietário
Suportadas MySQL sabores	MySQL , servidor Percona para MySQL , MariaDB , Percona XtraDB *cluster*, MariaDB Galera *cluster*	MySQL
Sistemas Operacionais Suportados	Linux	Linux, Solaris, Windows, OSX, FreeBSD.
Backups InnoDB sem bloqueio [1]	sim	sim
Bloqueando *backups* do MyISAM	sim	sim
Backups incrementais	sim	sim
Backups compactados completos	sim	sim
Backups compactados incrementais	sim	
Backups incrementais rápidos [2]	sim	
Backups incrementais com logs arquivados são apresentados no Percona Server	sim	
Backups incrementais apenas com log REDO		sim
Bloqueios de *backup* [8]	sim	

Tabela 42 - Funcionalidade de *backups* Percona SSTv2 vs MySQL [14] 2/3

Recursos	**Percona XtraBackup**	***Backup* do MySQL Enterprise**
Backups criptografados	sim	Sim [3]
Backup de *streaming*	sim	sim
Backups locais paralelos	sim	sim
Compressão paralela	sim	sim
Criptografia paralela	sim	sim
Apply-log paralelo	sim	
Cópia paralela		sim
Backups parciais	sim	sim
Backups parciais de partições individuais	sim	
Regulagem de pressão [4]	sim	sim
Validação de imagem de *backup*		sim
Suporte de recuperação *point-in-time*	sim	sim
Backups seguros de escravos	sim	
Backups compactos [5]	sim	
Backups de estado do buffer pool	sim	
Exportação de tabelas individuais	sim	Sim [6]
Exportação de partições individuais	sim	
Restaurando tabelas para um servidor diferente [7]	sim	sim

Tabela 43 - Funcionalidade de *backups* Percona SSTv2 vs MySQL [14] 3/3

Recursos	**Percona XtraBackup**	***Backup* do MySQL Enterprise**
Desfragmentação de índices secundários do InnoDB	sim	
rsync suporte para minimizar o tempo de bloqueio	sim	
FTWRLManuseio aprimorado	sim	
Tabela de histórico de *backup*	sim	sim
Tabela de progresso do *backup*		sim
Backups offline		sim
Backup para gerenciadores de mídia de fita		sim
Suporte para *backups* na nuvem		Amazon S3
Interfaces gráficas externas do usuário para *backup* / recuperação	Zmanda Recovery Manager para MySQL	MySQL Workbench, MySQL Enterprise Monitor

Anexo XII. Instalação e configuração do Percona XtraDB *cluster*

Nesta secção enumera-se o procedimento de instalação do Percona XtraDB *cluster* no sistema operativo Centos 7, descrevendo a parametrização dos pacotes necessários para que a aplicação possa ser posteriormente configurada de acordo com a arquitetura do presente projeto (Figura 37). A versão utilizada (5.7) para a instalação do PXC, não foi a versão 8.0 (atual) pois esta não estava estabilizada à data de implementação deste sistema de armazenamento.

O *cluster* adotado é composto por três sistemas informáticos, nos quais o PXC foi instalado com auxílio dos seguintes passos:

1. Configuração dos repositórios do Percona

```
$ sudo yum install https://repo.percona.com/yum/percona-release latest.noarch.rpm
```

2. Instalação dos pacotes do Percona XtraDB *cluster* (versão 5.7):

```
$ sudo yum install Percona-XtraDB-cluster-57
```

3. Iniciar o serviço `mysql` e colocar o processo permanente no arranque do sistema:

```
$ sudo systemctl start mysql
$ sudo systemctl enable mysql
```

4. Copiar a senha temporária gerada automaticamente para a conta do utilizador:

```
$ sudo grep 'temporary password' /var/log/mysqld.log
```

5. Utilizar a senha temporária para efetuar *login* pela primeira vez em modo root:

```
$ mysql -u root -p temporary password
```

6. Alterar a senha da conta local *root*:

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'rootPass';
Query OK, 0 rows affected (0.00 sec)

mysql> exit
```

7. Parar o serviço mysql

```
$ sudo service mysql stop
```

Configuração dos nós

A seguinte configuração e associação dos nós de Percona ao *cluster* compreende os endereços e nomenclaturas apresentadas na Tabela 44.

Tabela 44 - Terminologia dos nós para configuração do PXC

Identificação do *cluster*	Endereço IPv4	Nome do *host*
pxc-*cluster*	10.200.100.127	perconadb1
	10.200.100.128	Perconadb2
	10.200.100.129	Perconadb3

1. Diretorias de configuração do PXC

```
$ vi /etc/my.cnf

  !includedir /etc/my.cnf.d/
  !includedir /etc/percona-xtradb-cluster.conf.d/
```

2. Editar o ficheiro de configuração nos três nós do PXC:

```
$ vi/etc/percona-xtradb-cluster.conf.d/wsrep.cnf
```

3. Para cada nó foram realizadas as configurações (ficheiro `wsrep.cnf`) de acordo com o anexo XII e em consonância com a alteração dos seguintes campos genéricos:

```
[mysqld]
wsrep_cluster_address=gcomm://<ip_node1,ip_node2,ip_node3>

# Node IP address
wsrep_node_address=<ip_node_local>

# cluster name
wsrep_cluster_name=<cluster_name>

#If wsrep_node_name is not specified,  then system hostname will be used
wsrep_node_name= pxc-cluster

#pxc_strict_mode allowed values: DISABLED,PERMISSIVE,ENFORCING,MASTER
pxc_strict_mode=ENFORCING

# SST method
wsrep_sst_method=xtrabackup-v2

#Authentication for SST method
wsrep_sst_auth="sstuser:**secret-pass**"
```

4. Após configurar o passo 3 nos três nós, o primeiro nó a iniciar o `mysql` terá de ser o "*bootstrap node"*. O comando para iniciar o procedimento citado é o seguinte:

```
[root@perconadb1~]#  systemctl start mysql@bootstrap.service
```

5. Iniciar o processo `mysql` no segundo e terceto nó, para sincronizar o PXC:

```
[root@perconadb2~]#  systemctl start mysql
[root@perconadb3~]#  systemctl start mysql
```

Ajustes de desempenho do InnoDB

Descrevem-se ainda os ajustes de *performance* do *InnoDB* (mecanismo de armazenamento do Mysql) realizados nos três nós de computação alusivos ao PXC, sendo que estes valores foram ajustados mediante o período experimental de aproximadamente um ano com o sistema em produção: A instanciação e o ajuste dos valores, é definido no ficheiro `mysqld.cnf` (localizado em `/etc/percona-xtradb-cluster.conf.d/`).

```
# Template my.cnf for PXC
# Edit to your requirements.
[client]
socket=/var/lib/mysql/mysql.sock

[mysqld]
server-id=1
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
log-error=/var/log/mysqld.log
pid-file=/var/run/mysqld/mysqld.pid
log-bin
log_slave_updates
expire_logs_days=7

#dns skip
skip_name_resolve=ON

#thread efficiency
thread_handling=pool-of-threads
thread_pool_size=24
max_connections=1500

#innodb
innodb_buffer_pool_size=4294967296

# Disabling symbolic-links is recommended to prevent assorted security risks
symbolic-links=0
```

Figura 85 - Ajustes de desempenho do mecanismo InnoDB

Anexo XIII. Configuração do Percona XtraDB *cluster* em três nós de computação

Neste anexo são demonstrados os ficheiros de configuração dos três nós que decompõe o PXC. Por omissão o ficheiro vem maioritariamente parametrizado, sendo que as alterações realizadas estão destacadas em tom de preto**.**

1. Configuração do primeiro nó do PXC:

```
[mysqld]
# Path to Galera library
wsrep_provider=/usr/lib64/galera3/libgalera_smm.so

# cluster connection URL contains IPs of nodes
# If no IP is found, this implies that a new cluster needs to be created,
# in order to do that you need to bootstrap this node
wsrep_cluster_address=gcomm://10.200.100.127,10.200.100.128,10.200.100.129

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# Slave thread to use
wsrep_slave_threads= 8
wsrep_log_conflicts

# This changes how InnoDB autoincrement locks are managed and is a requirement for Galera
innodb_autoinc_lock_mode=2

# Node IP address
wsrep_node_address=10.200.100.127

# cluster name
wsrep_cluster_name=pxc-cluster

#If wsrep_node_name is not specified,  then system hostname will be used
wsrep_node_name=perconadb1

#pxc_strict_mode allowed values: DISABLED,PERMISSIVE,ENFORCING,MASTER
pxc_strict_mode=ENFORCING

# SST method
wsrep_sst_method=xtrabackup-v2

#Authentication for SST method
wsrep_sst_auth="sstuser:**secret-pass**"
```

2. Configuração do segundo nó do PXC:

```
[mysqld]
# Path to Galera library
wsrep_provider=/usr/lib64/galera3/libgalera_smm.so

# cluster connection URL contains IPs of nodes
# If no IP is found, this implies that a new cluster needs to be created,
# in order to do that you need to bootstrap this node
wsrep_cluster_address=gcomm://10.200.100.127,10.200.100.128,10.200.100.129

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# Slave thread to use
wsrep_slave_threads= 8
wsrep_log_conflicts

# This changes how InnoDB autoincrement locks are managed and is a requirement for Galera
innodb_autoinc_lock_mode=2

# Node IP address
wsrep_node_address=10.200.100.128

# cluster name
wsrep_cluster_name=pxc-cluster

#If wsrep_node_name is not specified,  then system hostname will be used
wsrep_node_name=perconadb2

#pxc_strict_mode allowed values: DISABLED,PERMISSIVE,ENFORCING,MASTER
pxc_strict_mode=ENFORCING

# SST method
wsrep_sst_method=xtrabackup-v2

#Authentication for SST method
wsrep_sst_auth="sstuser:**secret-pass**"
```

3. Configuração do terceiro nó do PXC:

```
[mysqld]
# Path to Galera library
wsrep_provider=/usr/lib64/galera3/libgalera_smm.so

# cluster connection URL contains IPs of nodes
# If no IP is found, this implies that a new cluster needs to be created,
# in order to do that you need to bootstrap this node
wsrep_cluster_address=gcomm://10.200.100.127,10.200.100.128,10.200.100.129

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# Slave thread to use
wsrep_slave_threads= 8
wsrep_log_conflicts

# This changes how InnoDB autoincrement locks are managed and is a requirement for Galera
innodb_autoinc_lock_mode=2

# Node IP address
wsrep_node_address=10.200.100.129

# cluster name
wsrep_cluster_name=pxc-cluster

#If wsrep_node_name is not specified,  then system hostname will be used
wsrep_node_name=perconadb3

#pxc_strict_mode allowed values: DISABLED,PERMISSIVE,ENFORCING,MASTER
pxc_strict_mode=ENFORCING

# SST method
wsrep_sst_method=xtrabackup-v2

#Authentication for SST method
wsrep_sst_auth="sstuser:**secret-pass**"
```

Anexo XIV. Instalação e configuração da aplicação HAproxy

Enumera-se neste subcapítulo a metodologia de configuração da aplicação HAproxy. A mesma compreende um processo simples de parametrização, fornecendo uma arquitetura de encaminhamento baseada na camada de rede e transporte.

Instalação do HAproxy

O modelo de *cluster proxy* escolhido é constituído por dois sistemas informáticos, nos quais foi instalado a aplicação HAproxy com auxílio dos seguintes passos:

1. Configuração dos repositórios do Centos 7 para atualizar disponibilizar novas versões de HAproxy:

```
$ sudo yum install epel-release
```

2. Instalação do HAproxy:

```
$ sudo yum install haproxy
```

3. Iniciar o serviço haproxy e colocar o processo permanente no arranque do sistema:

```
$ sudo systemctl start haproxy
$ sudo systemctl enable haproxy
```

Configuração do HAproxy

A parametrização do ficheiro de configuração (haproxy.cfg) do HAproxy foi realizada tendo em conta as especificações da solução, conforme demonstrado na Tabela 45.

Tabela 45 - Nomenclaturas de configuração para o HAproxy

Endereço IPv4	Nome do *host*	Aplicações	Encaminhamento HAproxy	
			Porta de *frontend*	Porta de *backend*
10.200.100.121	*proxy1*	HAproxy		6033
10.200.100.122	*proxy2*	HAproxy		6033
10.200.100.123	*proxy-cluster*	Pacemaker	3306	

O objetivo da configuração do HAproxy será apenas o reencaminhamento do tráfego para o proxySQL. Isto é, o HAproxy apenas irá redirecionar pedidos IP provenientes do endereço 10.200.100.123, na porta 3306 e na porta 9000 (gestão aplicacional *web*).

1. Localização do ficheiro de configuração do HAproxy em Centos 7:

```
$ sudo vi /etc/haproxy/haproxy.cfg
```

2. Parametrização do ficheiro de configuração haproxy.cfg:

```
#Content for inside haproxy
global
    log         127.0.0.1   local1
    log         127.0.0.1   local1 notice
    maxconn     8192
    user        haproxy
    group       haproxy
    nbproc      1
    pidfile     /var/run/haproxy.pid

defaults
    log         global
    option      tcplog
    option      dontlognull
    retries     3
    maxconn     4096
    option      redispatch
    timeout     connect 50000ms
    timeout     client 50000ms
```

Figura 86 - Configuração do ficheiro haproxy.cfg 1/3

```
#Content for inside haproxy
# defaults
    timeout     http-request 10s
    timeout     queue 1m
    timeout     connect 10s
    timeout     client 60m
    timeout     server 60m
    timeout     check 10s

# frontend (input)
frontend Zabbix_Server_Mysql
    bind 10.200.100.123:3306
    mode tcp
    option tcplog
    default_backend proxySQL
    maxconn 20000

# backend (output)
backend proxySQL
  balance leastconn
  mode tcp
  fullconn 2000
  stick store-request src
  stick-table type ip size 200k expire 30m
  hash-type consistent
    server proxySQL1 10.200.100.121:6033 check
    server proxySQL2 10.200.100.122:6033 check backup
```

Figura 87 - Configuração do ficheiro haproxy.cfg 2/3

```
#HAproxy web ui
frontend stats
    bind 10.200.100.123:9000
    mode http
    stats enable
    stats show-node
    stats uri /haproxy
    stats realm HAproxy\ Statistics
    stats auth haproxy:haproxy
    stats admin if TRUE
```

Figura 88 - Configuração do ficheiro haproxy.cfg 3/3

3. Localização e edição dos ficheiros de log:

```
#For Access Log (local1.info)
$ tail -f /var/log/haproxy-access.log

#For service Info, Backend and loadbalancer (local1.notice)
$ tail -f /var/log/haproxy-info.log
```

O ficheiro de configuração `haproxy.cfg` está dividido em cinco grandes blocos parametrizáveis

- *`global`*: resume-se à definição dos *logs*, utilizadores e grupos, bem como o número máximo de conceções que o HAproxy suporta;
- *`defaults`*: determina maioritariamente as funções de controlo de temporal (*timeout*) das ligações reencaminhadas;
- *`frontend`*: representa o ponto de acesso dos clientes. Especifica o parâmetro `bind`, o qual indica o conjunto de <IP:porta> onde o HAproxy recebe os dados para serem encaminhados ao módulo de `backend`. Podem ser ainda definidos dois modos de redireccionamento dos dados (tcp e http). Em relação ao modo tcp, este não aufere a validação de mensagens. Por sua vez, o modo http retransmite a informação com base na análise das mensagens http;
- *`backend`*: caracteriza um conjunto de servidores (`server`) para onde os dados são encaminhados. A forma de balanceamento (`balance`) de carga entre os nós é definida por quatro algoritmos comumente utilizados:
  - O algoritmo `roundrobin` adota metodologia de balanceamento de carga idêntica entre os servidores com critérios de “pesos”;

- Por outro lado, o `lastconn` elege o servidor com menor número de sessões ativas, sendo este aconselhado para sessões mais longas, tais como SQL, LDAP, TSE, entre outros;
- A escolha do servidor baseada num identificador numérico pela ordem do ID mais baixo para o mais alto, recorre ao algoritmo `first`;
- Por último, o algoritmo `source` distribui a carga com base no IP de origem do cliente;

- *`frontend stats`*: descreve as configurações de autenticação para acesso da plataforma *web*, bem como, o campo `bind` associa o <`IP:porta`> onde interface gráfica pode ser consultada. O campo `show node` demonstra o servidor do *cluster* HAproxy que está ativo.

Numa perspetiva global o ficheiro (`haproxy.cfg`) foi parametrizado de acordo com as configurações demonstradas, estando as principais parametrizações realizadas destacadas a "negrito".

É importante referir, que a instalação e configuração foi realizada nos dois servidores *proxy1* e *proxy2* (Tabela 45). A aplicação HAproxy responde a um único endereço VIP (10.200.100.123), e foi devidamente associada ao Pacemaker, de modo a garantir a indisponibilidade de um dos sistemas (Figura 46).

Anexo XV. Instalação e configuração da aplicação proxySQL

No que diz respeito ao proxySQL, este necessita de um entendimento teórico, de modo a proceder à sua configuração mais personalizada e detalhada os requisitos da solução. Salienta-se que o proxySQL complementa o encaminhamento baseado na camada aplicacional, possibilitando o controlo remoto do PXC.

Arquitetura e funcionamento:

Toda a configuração do proxySQL recai sobre uma arquitetura modelar, dividida em quatro camadas fundamentais:

- Configuração: destaca-se como a camada inicial que representa o ficheiro de configuração (`proxysql.cnf`);
- Memória: esta camada não representa a configuração em execução, define apenas uma base de dados auxiliar e mantida pelo proxySQL para as suas funcionalidades;
- Disco: retrata as configurações guardadas, sendo estas armazenadas numa base de dados SQLlite;
- Execução: constitui uma camada que efetiva as configurações utilizadas pelo proxySQL em tempo real.

A configuração é realizada no ficheiro `proxysql.cnf` ou através do módulo admin. O fluxo de todas as alterações realizadas no ficheiro de configuração é representado na Figura 89.

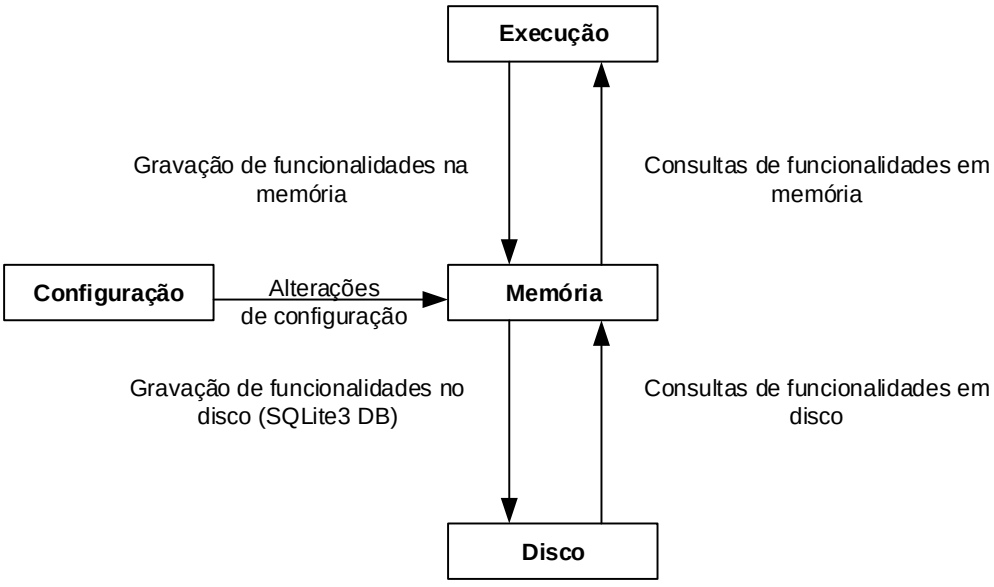


Figura 89 - Fluxos de configuração entre camadas do proxySQL

As funcionalidades provenientes de fluxos de configuração, de acordo com a Figura 89, são demonstradas na Tabela 46

Tabela 46 - Principais funcionalidades de configuração de proxySQL

Funcionalidades (comandos)		Descrição
Principais tabelas de gestão MySQL	mysql_users	Contém a lista de utilizadores para autenticação
	mysql_servers	Inclui a lista de servidores e respetivos HG (Host Group)
	mysql_query_rules	Define a lista de regras para a cache, nomeadamente, reescrita e redireccionamento de *queries*.
Gestão MySQL	mysql variables	Variáveis que controlam as funcionalidades MySQL através do proxySQL
Gestão administrativa	admin variables	Variáveis que determinam a gestão administrativa através da *command line* interface (porta 6032).

Todas as funcionalidades do proxySQL, englobam métodos (*mysql variables* e *admin variables*) que permitem fazer a gestão da base de dados alojada no *cluster* (PXC), através da linha de comandos disponibilizada pelo proxySQL.

Instalação do proxySQL2

A instalação do proxySQL foi realizada em dois sistemas computacionais distintos, nos quais, foram previamente instaladas as ferramentas de HAproxy, de modo a integrar um mecanismo de encaminhamento para os nós de proxySQL (Figura 41). Enumeram-se os seguintes passos:

1. Instalação do proxySQL2 em Centos 7:

```
$ sudo yum install proxysql2
```

2. Instalação alternativa do proxySQL2 (versão 2.0.10):

```
$ sudo yum install
https://www.percona.com/downloads/proxysql2/proxysql22.0.10/binary/redhat/7/x86_64/prox
ysql2-2.0.10-1.1.el7.x86_64.rpm
```

3. Iniciar o serviço `proxysql` e colocar o processo permanente no arranque do sistema:

```
$ sudo systemctl start proxysql
$ sudo systemctl enable proxysql
```

Configuração do proxySQL2

O modo de configuração advém dos objetivos da solução proposta. Com auxílio da Tabela 47 fica percetível as portas de encaminhamento que serão utilizadas pela aplicação proxySQL, bem como os sistemas computacionais que irão se enquadrar nessa tarefa.

Tabela 47 - Especificações para configuração do proxySQL

Endereço IPv4	Nome do *host*	Aplicações	Encaminhamento proxySQL	
			Porta *Inbound*	Porta *outbound*
10.200.100.121	*proxy1*	proxySQL2	6033	3306
10.200.100.122	*proxy2*	proxySQL2	6033	3306
10.200.100.123	*proxy-cluster*	Haproxy	3306	6033

Antes de começar a parametrizar o proxySQL, por motivos de segurança, é conveniente alterar as palavras-chave das quatro contas (Tabela 48) especificadas no ficheiro de configuração `admin.cnf` (Figura 104).

Tabela 48 - Contas padrão da ferramenta proxySQL

Contas MySQL locais/remotas	Utilizador	Palavra-chave	IP do nó de acesso	Porta de acesso
Interface administrativa do proxySQL (local)	admin	admin	localhost	6032
Interface administrativa do PXC (remota)	admin	admin	localhost	6033
Monitorização do PXC (remota)	monitor	monit0r		
Interligação do proxySQL com o PXC (remota)	proxysql_user	passw0rd		

A metodologia utilizada teve por base o configurador automático (ver Anexo XVI), o qual recorre ao *script* `proxysql-admin` e às configurações pré-definidas no ficheiro `admin.cnf`. Com a funcionalidade de configuração automática, surge então a possibilidade de utilizar comados genéricos, capazes de simplificar o processo de configuração.

Enumeram-se de seguida, os principais comandos para configurar o proxySQL de modo a controlar os nós do PXC:

- `--enable`
- `--adduser`
- `--syncusers`
- `--add-query-rule`
- `--quick-demo`
- `--update-cluster`
- `--status`
- `--force`

Localização do ficheiro `proxysql-admin.cnf` para alteração de credenciais. É obrigatório modificar o nome do utilizador e a respetiva palavra-chave da conta associada à interface administrativa do PXC, de acordo com as credenciais de acesso à base dade dados do *cluster*:

```
$ sudo vi /etc/proxysql-admin.cnf

# PXC admin credentials for connecting to pxc-cluster-node.
export CLUSTER_USERNAME='proxysql'
export CLUSTER_PASSWORD='PXC'
export CLUSTER_HOSTNAME='10.200.100.127'
export CLUSTER_PORT='3306'
```

1. Criação de utilizadores “proxysql” (inerente aos dois sistemas do proxySQL) num dos nós do PXC (replicação multi-*master*):

```
$ [root@PXC1 ~]#  mysql -uroot -p <root-pxc-password>

mysql> CREATE USER 'proxysql'@'10.200.100.121' IDENTIFIED BY 'PXC';
mysql> CREATE USER 'proxysql'@'10.200.100.122' IDENTIFIED BY 'PXC';
mysql> GRANT ALL PRIVILEGES ON * . * TO 'proxysql'@'10.200.100.121';
mysql> GRANT ALL PRIVILEGES ON * . * TO 'proxysql'@'10.200.100.122';
mysql> FLUSH PRIVILEGES;
mysql> select user, host from mysql.user;
+------------------+----------------+
| user             | host           |
+------------------+----------------+
| proxysql         | 10.200.100.121 |
| proxysql         | 10.200.100.122 |
+------------------+----------------+
2 rows in set (0.11 sec)
```

2. Ativar (`--enable`) o script proxysql-admin para iniciar a configuração automática:

```
$ sudo proxysql-admin --config-file=/etc/proxysql-admin.cnf --enable
```

Figura 90 - Configuração automática do proxySQL 1/2

```
This script will assist with configuring proxySQL for use with
Percona XtraDB cluster (currently only PXC in combination
with proxySQL is supported)

proxySQL read/write configuration mode is singlewrite

Configuring the proxySQL monitoring user.
proxySQL monitor user name as per command line/config-file is monitor

User 'monitor'@'127.%' has been added with USAGE privileges

Configuring the Percona XtraDB cluster application user to connect through proxySQL
Percona XtraDB cluster application user name as per command line/config-file is
cluster_one

Percona XtraDB cluster application user 'proxysql_user'@'127.%' has been added with ALL
privileges, this user is created for testing purposes
Adding the Percona XtraDB cluster server nodes to proxySQL

Write node info
+----------------+--------------+--------+-----------+
|    hostname    | hostgroup_id |  port  |   weight  |
+----------------+--------------+--------+-----------+
| 10.200.100.128 | 10           |  3306  |  1000000  |
+----------------+--------------+--------+-----------+

proxySQL configuration completed!

proxySQL has been successfully configured to use with Percona XtraDB cluster

You can use the following login credentials to connect your application through proxySQL
mysql --user=proxysql_user -p --host=127.0.0.1 --port=6033 --protocol=tcp
```

Figura 91 - Configuração automática do proxySQL 2/2

3. Validar se os nós do PXC estão associados ao proxySQL:

```
$ mysql --user=proxysql_user -p --host=127.0.0.1 --port=6033 --protocol=tcp
mysql> select hostgroup_id,hostname,port,status from runtime_mysql_servers;
+--------------+----------------+------+--------+
| hostgroup_id | hostname       | port | status |
+--------------+----------------+------+--------+
| 10           | 10.200.100.128 | 3306 | ONLINE |
| 11           | 10.200.100.129 | 3306 | ONLINE |
| 11           | 10.200.100.127 | 3306 | ONLINE |
| 12           | 10.200.100.129 | 3306 | ONLINE |
| 12           | 10.200.100.127 | 3306 | ONLINE |
+--------------+----------------+------+--------+
```

4. Verificar e mapear o identificador de *hostgroup* via proxySQL:

```
mysql> select * from mysql_galera_hostgroups\G
*************************** 1. row ***************************
       writer_hostgroup: 10
backup_writer_hostgroup: 12
       reader_hostgroup: 11
      offline_hostgroup: 13
                 active: 1
            max_writers: 1
  writer_is_also_reader: 2
max_transactions_behind: 100
```

5. Alterar o *default hostgroup* dos utilizadores MySQL para que estes possam ter privilégios de escrita:

```
$ update mysql_users set default_hostgroup = 10;
```

6. Verificar informações sobre o *cluster* Galera:

```
$ sudo proxysql-admin --status --writer-hg=10

mysql_galera_hostgroups row for writer-hostgroup: 10
+--------+--------+--------------+---------+--------+-------------+----------------------+------------------+
| writer | reader | backup-writer | offline | active | max_writers | writer_is_also_reader | max_trans_behind |
+--------+--------+--------------+---------+--------+-------------+----------------------+------------------+
| 10     | 11     | 12           | 13      | 1      | 1           | 2                    | 100              |
+--------+--------+--------------+---------+--------+-------------+----------------------+------------------+

mysql_servers rows for this configuration

+---------------+-------+----------------+------+--------+--------+----------+---------+---------+
| hostgroup     | hg_id | hostname       | port | status | weight | max_conn | use_ssl | gtid_port
+---------------+-------+----------------+------+--------+--------+----------+---------+---------+
| writer        | 10    | 10.200.100.128 | 3306 | ONLINE | 1000000| 1000     | 0       | 0        |
| reader        | 11    | 10.200.100.127 | 3306 | ONLINE | 1      | 1000     | 0       | 0        |
| reader        | 11    | 10.200.100.129 | 3306 | ONLINE | 1      | 1000     | 0       | 0        |
| backup-writer | 12    | 10.200.100.127 | 3306 | ONLINE | 1      | 1000     | 0       | 0        |
| backup-writer | 12    | 10.200.100.129 | 3306 | ONLINE | 1      | 1000     | 0       | 0        |
```

7. Validar todas as métricas e estatísticas de encaminhamento do proxySQL:

```
$ proxysql-status <user_admin> <pass_admin> 127.0.0.1 6032
```

Criação de utilizadores MySQL via proxySQL

Os utilizadores MySQL podem ser criados a partir do proxySQL. Neste caso, será dado um exemplo para um possível caso de uso futuro.

1. Criar utilizador na tabela mysql_users do PXC:

```
root@proxysql:~# mysql -u admin -p -P 6032 -h 127.0.0.1 --prompt='Admin> '
Admin> INSERT INTO mysql_users (username,password) VALUES ('teste','pass');
Query OK, 1 row affected (0.00 sec)
```

2. Atualizar as configurações realizadas para executar em tempo real e gravar no disco local do proxySQL:

```
Admin> LOAD MYSQL USERS TO RUNTIME;
Admin> SAVE MYSQL USERS TO DISK;
```

3. Permitir o acesso a escrita e leitura aos nós do PXC

```
$ [root@PXC1 ~]# mysql -uroot -p <root-pxc-password>

mysql> CREATE USER 'teste'@'10.200.100.121' IDENTIFIED BY 'pass';
mysql> GRANT ALL PRIVILEGES ON * . * TO 'teste'@'10.200.100.121';
mysql> FLUSH PRIVILEGES;
```

Anexo XVI. Configuração automática do proxySQL

A configuração automática do ProxySQL é fundamentada num conjunto de comandos, que permitem a parametrização de forma autónoma, através da interface de gestão. A premissa `proxysql-admin` permite utilizar diversos comandos para gerir o PXC, bem como os métodos de encaminhamentos dos dados.

```
Usage: proxysql-admin [ options ]

Options:

--config-file=<config-file>        Read login credentials from a configuration file
                                   (command line options override any configuration file values)

--writer-hg=<number>               The hostgroup that all traffic will be sent to
                                   by default. Nodes that have 'read-only=0' in MySQL
                                   will be assigned to this hostgroup.
--backup-writer-hg=<number>        If the cluster has multiple nodes with 'read-only=0'
                                   and max_writers set, then additional nodes (in excess
                                   of max_writers), will be assigned to this hostgroup.
--reader-hg=<number>               The hostgroup that read traffic should be sent to.
                                   Nodes with 'read-only=0' in MySQL will be assigned
                                   to this hostgroup.
--offline-hg=<number>              Nodes that are determined to be OFFLINE will
                                   assigned to this hostgroup.

--proxysql-datadir=<datadir>       Specify the proxysql data directory location
--proxysql-username=<user_name>    proxySQL service username
--proxysql-password[=<password>]   proxySQL service password
--proxysql-port=<port_num>         proxySQL service port number
--proxysql-hostname=<host_name>    proxySQL service hostname

--cluster-username=<user_name>     Percona XtraDB cluster node username
--cluster-password[=<password>]    Percona XtraDB cluster node password
--cluster-port=<port_num>          Percona XtraDB cluster node port number
--cluster-hostname=<host_name>     Percona XtraDB cluster node hostname

--cluster-app-username=<user_name> Percona XtraDB cluster node application username
--cluster-app-password[=<password>] Percona XtraDB cluster node application passwrod
--without-cluster-app-user         Configure Percona XtraDB cluster without application user

--monitor-username=<user_name>      Username for monitoring Percona XtraDB cluster nodes through
proxySQL
--monitor-password[=<password>]     Password for monitoring Percona XtraDB cluster nodes through
proxySQL
--use-existing-monitor-password    Do not prompt for a new monitor password if one is provided.

--node-check-interval=<NUMBER>     The interval at which the proxy should connect
                                   to the backend servers in order to monitor the
                                   Galera staus of a node (in milliseconds).
                                   (default: 5000)
--mode=[loadbal|singlewrite]       proxySQL read/write configuration mode
                                   currently supporting: 'loadbal' and 'singlewrite'
                                   (default: 'singlewrite')
--write-node=<IPADDRESS>:<PORT>    Specifies the node that is to be used for
                                   writes for singlewrite mode.  If left unspecified,
                                   the cluster node is then used as the write node.
                                   This only applies when 'mode=singlewrite' is used.
--max-connections=<NUMBER>         Value for max_connections in the mysql_servers table.
                                   This is the maximum number of connections that
                                   proxySQL will open to the backend servers.
                                   (default: 1000)
```

Figura 92 - Comandos de configuração automática do proxySQL2 1/2

```
--max-transactions-behind=<NUMBER> Determines the maximum number of writesets a node
                                   can have queued before the node is SHUNNED to avoid
                                   stale reads.
                                   (default: 100)
--use-ssl=[yes|no]                If set to 'yes', then connections between proxySQL
                                   and the backend servers will use SSL.
                                   (default: no)
--writers-are-readers=[yes|no|backup]
                                   If set to 'yes', then all writers (backup-writers also)
                                   are added to the reader hostgroup.
                                   If set to 'no', then none of the writers (backup-writers also)
                                   will be added to the reader hostgroup.
                                   If set to 'backup', then only the backup-writers
                                   will be added to the reader hostgroup.
                                   (default: backup)
--remove-all-servers              When used with --update-cluster, this will remove all
                                   servers belonging to the current cluster before
                                   updating the list.
--debug                           Enables additional debug logging.
--help                            Dispalys this help text.

These options are the possible operations for proxysql-admin.
One of the options below must be provided.
--adduser                         Adds the Percona XtraDB cluster application user to the proxySQL
database
--disable, -d                     Remove any Percona XtraDB cluster configurations from proxySQL
--enable, -e                      Auto-configure Percona XtraDB cluster nodes into proxySQL
--update-cluster                  Updates the cluster membership, adds new cluster nodes
                                   to the configuration.
--update-mysql-version            Updates the `mysql-server_version` variable in proxySQL with the
version
                                   from a node in the cluster.
--quick-demo                      Setup a quick demo with no authentication
--syncusers                       Sync user accounts currently configured in MySQL to proxySQL
                                   May be used with --enable.
                                   (deletes proxySQL users not in MySQL).
                                   See:ref:`pxc.proxysql.v2.admin-tool.syncusers`     for     more
information
--sync-multi-cluster-users        Sync user accounts currently configured in MySQL to proxySQL
                                   May be used with --enable.
                                   (doesn't delete proxySQL users not in MySQL)
--add-query-rule                  Create query rules for synced mysql user. This is applicable only
                                   for singlewrite mode and works only with --syncusers
                                   and --sync-multi-cluster-users options
--is-enabled                      Checks if the current configuration is enabled in proxySQL.
--status                          Returns a status report on the current configuration.
                                   If "--writer-hg=<NUM>" is specified, than the
                                   data corresponding to the galera cluster with that
                                   writer hostgroup is displayed. Otherwise, information
                                   for all clusters will be displayed.
--force                                  This option will skip existing configuration checks in
mysql_servers,
                                   mysql_users and mysql_galera_hostgroups tables. This option will
only
                                   work with ``proxysql-admin --enable``.
--disable-updates                 Disable admin updates for proxySQL cluster for the
                                   current operation. The default is to not change the
                                   admin variable settings.  If this option is specifed,
                                   these options will be set to false.
                                   (default: updates are not disabled)
--version, -v                     Prints the version info
```

Figura 93 - Comandos de configuração automática do proxySQL2 2/2

Anexo XVII. Configuração da ferramenta Keepalived nos servidores de *proxy*

Nos dois sistemas integrantes do *cluster* de *proxys* foram instalados e parametrizados com a aplicação Keepalived, de acordo com os seguintes passos:

1. Instalação do Keepalived em Centos 7 (versão 1.3.5-16):

```
$ sudo yum install keepalived
```

2. Iniciar o serviço `keepalived` e colocar o processo permanente no arranque do sistema:

```
$ sudo systemctl start keepalived
$ sudo systemctl enable keepalived
```

3. Parametrizar o *kernel* do sistema operativo para suportar endereço IP virtual:

```
$ echo "net.ipv4.ip_nonlocal_bind = 1" >> /etc/sysctl.conf
```

4. Permissão para o scritp do Keepalived aceder as aplicações do sistema:

```
$ chkconfig keepalived on
```

5. Editar o ficheiro de configurações nos dois sistemas computacionais e seguir as parametrizações apresentadas na:

```
$ vi /etc/keepalived/keepalived.conf
```

```
vrrp_script chk_haproxy {
  script "killall -0 haproxy" # check the haproxy process
  interval 2 # every 2 seconds
  weight 2 # add 2 points if OK
}

vrrp_instance VI_1 {
  interface eth0 # interface to monitor
  state MASTER # MASTER on haproxy1, BACKUP on haproxy2
  virtual_router_id 51
  priority 101 # 101 on haproxy1, 100 on haproxy2
  virtual_ipaddress {
    10.200.100.123 # virtual ip address
  }
  track_script {
    chk_haproxy
  }
}
```

Figura 94 - Configuração Keepalived no servidor *proxy* 1

```
vrrp_script chk_haproxy {
  script "killall -0 haproxy" # check the haproxy process
  interval 2 # every 2 seconds
  weight 2 # add 2 points if OK
}

vrrp_instance VI_1 {
  interface eth0 # interface to monitor
  state MASTER # MASTER on haproxy1, BACKUP on haproxy2
  virtual_router_id 51
  priority 100 # 101 on haproxy1, 100 on haproxy2
  virtual_ipaddress {
    10.200.100.123 # virtual ip address
  }
  track_script {
    chk_haproxy
  }
}
```

Figura 95 - Configuração Keepalived no servidor *proxy* 2

<u>Comandos para executar testes</u>:

1. Verificar o IP virtual no sistema:

```
$ ip a
```

2. Averiguar qual das máquinas está definida como *master/backup*

```
$ tail -f /var/log/messages
```

Anexo XVIII. Configuração da ferramenta Pacemaker nos servidores de *proxy*

Nos dois sistemas computacionais do *cluster* de *proxys* foram instalados e parametrizados com a aplicação Pacemaker, de acordo com os seguintes passos:

1. Instalação do pacote "Pacemaker" em Centos 7 (versão: 1.1.20-5.el7_7.2):

```
$ sudo yum install pacemaker
```

2. Instalação do pacote "PCS" em Centos 7 (versão: 0.9.167-3.el7):

```
$ sudo yum install pcs
```

3. Nos dois servidores de *proxy*, foram configurados os seguintes utilizadores e as respetivas palavras-chave para a aplicação PCS:

```
#proxy1 e proxy2
$ sudo passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

4. Iniciar o serviço `pcsd` e colocar o processo permanente no arranque do sistema:

```
$ sudo systemctl start pcsd.service
$ sudo systemctl enable pcsd.service
```

5. Mapear os endereços IP dos servidores em nomes (editar o ficheiro em ambos os sistemas):

```
#proxy1 e proxy2
$ sudo vi /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

10.200.100.121 proxy1
10.200.100.122 proxy2
```

6. Configuração do *cluster* pcs (configuração realizada em ambos os servidores), para agregar os processos a serem controlados com base no Corosync:

```
#autenticação
 pcs cluster auth proxy1 proxy2
 username: hacluster
 Password: <secret pass>

#criação do cluster proxy
pcs cluster setup --local --name cluster_proxy proxy1 proxy2

#iniciar cluster
pcs cluster start --all
```

7. Colocar os serviços `corosync` e `pacemacker` a arrancar automaticamente na inicialização do sistema em ambos os servidores *proxy*:

```
$ sudo systemctl enable corosync.service
$ sudo systemctl enable pacemaker.service
```

8. Por predefinição o modo de segurança dos dados vem ativado, assim sendo, na fase inicial de partilha dos dados entre os sistemas que decompõe o *cluster* de proxys, deve-se desabilitar o mecanismo stonith:

```
#proxy1 e proxy2
$ sudo pcs property set stonith-enabled=false
```

9. Desativar o mecanismo de quorum, pois num modelo de *cluster* com dois sistemas computacionais, não é possível utilizar este modelo (mínimo de 3 servidores):

```
#proxy1 e proxy2
$ pcs property set no-quorum-policy=ignore
```

10. Associação do endereço IP virtual (configurar em apenas um servidor):

```
$ pcs resource create cluster_vip ocf:heartbeat:IPaddr2 ip=10.200.100.123 cidr_netmask=24
op monitor interval=3s
```

11. Comandos para testar o *cluster* de *proxys* e os respetivos nós constituintes:

```
$ pcs status cluster

cluster Status:
 Stack: corosync
 Current DC: proxy1 (version 1.1.20-5.el7_7.2-3c4c782f70) - partition with quorum
 Last updated: Sat Jun 20 19:43:30 2020
 Last change: Wed May 27 21:09:02 2020 by root via cibadmin on proxy2
 2 nodes configured
 3 resources configured

$ pcs status nodes

Pacemaker Nodes:
 Online: proxy1 proxy2
 Standby:
 Maintenance:
 Offline:
Pacemaker Remote Nodes:
 Online:
 Standby:
 Maintenance:
 Offline:
```

12. Desativar o serviço `haproxy` nos dois servidores de *proxy*:

```
#proxy1 e proxy2
$ systemctl disable haproxy
```

13. Possibilitar a gestão do serviço `haproxy` através do Pacemaker (configurar apenas num dos servidores):

```
$ pcs resource create haproxy systemd:haproxy op monitor interval=3s
```

14. Associar o serviço `haproxy` ao endereço VIP do *cluster* de *proxys*:

```
$ pcs constraint colocation add haproxy cluster_vip
```

15. Ordenar a prioridade de inicialização de um dado serviço, concretamente neste cenário só foi adicionado a aplicação `haproxy`:

```
$ pcs constraint order cluster_vip then haproxy
```

16. Definir de forma estática, qual dos dois servidores do *cluster* é o ativo ou o passivo. Neste caso foi definido o servidor proxy1 como servidor primário:

```
$ pcs constraint location cluster_vip then haproxy1
$ pcs constraint location haproxy then haproxy1
```

17. Inicializar o serviço Pacemaker:

```
$ service pacemaker start
```

Anexo XIX. Instalação e configuração do *cluster* de servidores Zabbix

Com auxílio da Tabela 49, podemos perceber as especificações gerais dos sistemas computacionais a serem parametrizados, bem como os pacotes aplicacionais a serem instalados em cada servidor Zabbix. Ambos os servidores irão conter as mesmas configurações, formando assim, um *cluster* de alta disponibilidade através da ferramenta Pacemaker.

Tabela 49 - Especificações para configuração dos servidores Zabbix

Endereço IPv4	Nome do *host*	Aplicações	Componente
10.200.100.111	Zabbix1	zabbix-server-mysql zabbix-web-mysql pacemaker	Zabbix *cluster*
10.200.100.112	Zabbix2	zabbix-server-mysql zabbix-web-mysql pacemaker	
10.200.100.113	VIP zabbix-*cluster*	pacemaker	

Foram seguidos os seguintes passos genéricos para instalar o Zabbix em ambos os sistemas informáticos:

1. Instalação do repositório Zabbix (versão: 4.4.3-1.el7) em Centos 7:

```
# zabbix1 e zabbix2
$ rpm -Uvh https://repo.zabbix.com/zabbix/4.4/rhel/7/x86_64/zabbix-release-4.4-
1.el7.noarch.rpm
$ yum clean all
```

2. Instalação dos pacotes para o servidor Zabbix, *front-end* e agente de monitorização (pacotes de instalação para suportar base de dados MySQL):

```
# zabbix1 e zabbix2
$ yum install zabbix-server-mysql zabbix-web-mysql zabbix-agent
```

3. Num dos nós do Percona XtraDB *cluster* previamente configurados (ver Anexo XIII), foi realizada a criação dos utilizadores MySQL referentes a cada servidor Zabbix, e à base de dados.

```
$ [root@PXC1 ~]# mysql -uroot -p <root-pxc-password>

mysql> create database zabbixdb character set utf8 collate utf8_bin;
mysql> CREATE USER 'zabbix_server'@'10.200.100.121' IDENTIFIED BY 'passw0rd';
mysql> CREATE USER 'zabbix_server'@'10.200.100.122' IDENTIFIED BY 'passw0rd';
mysql> grant all privileges on zabbixdb.* to 'zabbix_server'@'10.200.100.121';
mysql> grant all privileges on zabbixdb.* to 'zabbix_server'@'10.200.100.122';
mysql> FLUSH PRIVILEGES;
mysql> quit;
```

É importante referir que o *cluster* PXC recebe dados do *cluster* de *proxys*, e por isso os endereços configurados para os utilizadores "zabbix_server" são referentes ao endereço IP de cada instância de servidores proxy (ver Figura 36). Deste modo a ligação MySQL é sempre realizada através de dois intermediários de encaminhamento aplicacional (haproxy e proxysql) quando os servidores Zabbix necessitam de guardar ou alterar dados no *cluster* de armazenamento.

4. Importar o esquema de dados e tabelas do zabbix, para a base de dados "zabbixdb":

```
# zabbix1 ou zabbix2
$ zcat /usr/share/doc/zabbix-server-mysql*/create.sql.gz| mysql zabbixdb uzabbix_server
-h 10.200.100.123
```

O script `create.sql.gz` foi modificado com um incremento de chaves primárias nas tabelas que não continham esse campo. O modelo *multi-master* (Replication Galera) incrementado nos nós do PXC obriga a que todas as tabelas criadas pelo *script* sejam apresentadas por um identificador, de forma a validar a consistência entre os dados trocados pelos nós de armazenamento (ver secção 5.6).

5. Configuração do processo de autenticação (`DBUser`, `DBPassword`) para ligação à base de dados (`DBName`) nos servidores Zabbix:

```
# zabbix1 e zabbix2
$ sudo vi /etc/zabbix/zabbix_server.conf

DBHost=10.200.100.123
DBName=zabbixdb
DBUser=zabbix_server
DBPassword=passw0rd
```

Do ponto de vista dos servidores Zabbix, o endereço IP da base de dados (`DBHost`) é identificado pelo endereço VIP do *cluster* de *proxys* anteriormente parametrizado com auxílio da aplicação Pacemaker (Anexo XVIII).

6. Já com a solução em produção, verificou-se o seguinte erro, devendo este ser corrigido neste passo de configuração inicial dos servidores Zabbix:

```
# num dos nós do PXC, consultar logs:

$ [root@PXC1 ~]# tail -f /var/log/mysqld.log

ERROR 1105 (HY000): Percona-XtraDB-cluster prohibits use of DML command on a table
(zabbix.dbversion) without an explicit primary key with pxc_strict_mode = ENFORCING or
MASTER
```

Figura 96 - Incompatibilidade do *scritp* mysql do sistema Zabbix

À semelhança do ponto quatro, deparou-se com um problema semelhante, faltando identificadores de chaves primárias, desta vez em tabelas de controlo de consistências do Percona XtraDB *cluster* e do Galera ao qual este recorre nativamente. A solução passou por num dos nós do PXC, adicionar as seguintes configurações:

```
$ [root@PXC1 ~]# mysql -uroot -p <root-pxc-password>
mysql> alter table dbversion ADD id4galera INT PRIMARY KEY AUTO_INCREMENT;
mysql> alter table history add id4galera int key auto_increment;
mysql> alter table history drop primary key , add primary key (id4galera,clock);
mysql> alter table history_uint add id4galera BIGINT key auto_increment;
mysql> alter table history_uBIGINT drop primary key , add primary key (id4galera,clock);
mysql> alter table history_log add id4galera BIGINT key auto_increment;
mysql> alter table history_log drop primary key , add primary key (id4galera,clock);
mysql> alter table history_text add id4galera BIGINT key auto_increment;
mysql> alter table history_text drop primary key , add primary key (id4galera,clock);
mysql> alter table history_str add id4galera BIGINT key auto_increment;
mysql> alter table history_str drop primary key , add primary key (id4galera,clock)
```

7. Modificar o arquivo PHP, descomentando e alterando o fuso horário por definição:

```
# zabbix1 e zabbix2
$ sudo vi /etc/httpd/conf.d/zabbix.conf
php_value date.timezone Europe/Lisbon
```

8. Configurar o endereço IP de origem para que o tráfego de saída dos servidores Zabbix seja identificado de forma única. Concretamente o servidor que tiver o endereço VIP ativo, será aquele que irá responder aos “pedidos”:

```
# zabbix1 e zabbix2
$ sudo vi /etc/zabbix/zabbix_server.conf

### Option: SourceIP
# Source IP address for outgoing connections.
# Mandatory: no
# Default:
SourceIP=10.200.100.113
```

9. Instalar e configurar a aplicação de Pacemaker em ambos os servidores Zabbix, de modo a associar os dois sistemas computacionais a um *cluster* de alta disponibilidade (consultar Anexo XX).
10. Iniciar os processos do servidor e Zabbix associados ao Pacemaker:

```
# zabbix1 e zabbix2
$ service pacemaker start
```

11. Acesso à configuração do Zabbix via ambiente gráfico (GUI, *Graphical User Interface*):

```
$ http://server_ip_or_name/zabbix
```

Anexo XX. Configuração da ferramenta Pacemaker nos servidores Zabbix

Nos dois sistemas computacionais do *cluster* de Zabbix foram instalados e parametrizados com a aplicação Pacemaker, de acordo com os seguintes passos:

1. Instalação do pacote "Pacemaker" em Centos 7 (versão: 1.1.20-5.el7_7.2):

```
$ sudo yum install pacemaker
```

2. Instalação do pacote "PCS" em Centos 7 (versão: 0.9.167-3.el7):

```
$ sudo yum install pcs
```

3. Nos dois servidores de Zabbix, foram configurados os seguintes utilizadores e as respetivas palavras-chave para a aplicação PCS:

```
#Zabbix1 e Zabbix2
$ sudo passwd hacluster
Changing password for user hacluster.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

4. Iniciar o serviço `pcsd` e colocar o processo permanente no arranque do sistema:

```
$ sudo systemctl start pcsd.service
$ sudo systemctl enable pcsd.service
```

5. Mapear os endereços IP dos servidores em nomes (editar o ficheiro em ambos os sistemas):

```
# zabbix1 e zabbix2
$ sudo vi /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

10.200.100.111 zabbix1
10.200.100.112 zabbix2
```

6. Configuração do *cluster* pcs (configuração realizada em ambos os servidores), para agregar os processos a serem controlados com base no Corosync:

```
#autenticação
 pcs cluster auth zabbix1 zabbix2
 username: hacluster
 Password: <secret pass>

#criação do cluster Zabbix
pcs cluster setup --local --name cluster_zabbix zabbix1 zabbix2

#iniciar cluster
pcs cluster start --all
```

7. Colocar os serviços `corosync` e `pacemacker` a arrancar automaticamente na inicialização do sistema em ambos os servidores Zabbix:

```
$ sudo systemctl enable corosync.service
$ sudo systemctl enable pacemaker.service
```

8. Por predefinição o modo de segurança dos dados vem ativado, assim sendo, na fase inicial de partilha dos dados entre os sistemas que decompõe o *cluster* de Zabbix, deve-se desabilitar o mecanismo stonith:

```
#zabbix1 e zabbix2
$ sudo pcs property set stonith-enabled=false
```

9. Desativar o mecanismo de quorum, pois num modelo de *cluster* com dois sistemas computacionais, não é possível utilizar este modelo (mínimo de 3 servidores):

```
#zabbix1 e zabbix2
$ pcs property set no-quorum-policy=ignore
```

10. Associação do endereço IP virtual (configurar em apenas um servidor):

```
$ pcs resource create cluster_vip ocf:heartbeat:IPaddr2 ip=10.200.100.113 cidr_netmask=24
op monitor interval=3s
```

11. Comandos para testar o *cluster* de Zabbix e os respetivos nós constituintes:

```
$ sudo pcs status cluster

cluster Status:
 Stack: corosync
 Current DC: proxy1 (version 1.1.20-5.el7_7.2-3c4c782f70) - partition with quorum
 Last updated: Sat Jun 20 19:43:30 2020
 Last change: Wed May 27 21:09:02 2020 by root via cibadmin on zabbix2
 2 nodes configured
 3 resources configured

$ sudo pcs status nodes

Pacemaker Nodes:
 Online: zabbix1 zabbix2
 Standby:
 Maintenance:
 Offline:
Pacemaker Remote Nodes:
 Online:
 Standby:
 Maintenance:
 Offline:
```

12. Desativar o serviço `zabbix-server` e `httpd` nos dois servidores de Zabbix:

```
#proxy1 e proxy2
$ systemctl disable zabbix-server
$ systemctl disable httpd
```

13. Possibilitar a gestão dos serviços através do Pacemaker (configurar apenas num dos servidores):

```
$ pcs resource create zabbix-server systemd: Zabbix-server op monitor interval=3s
$ pcs resource create httpd systemd: httpd op monitor interval=3s
```

14. Associar os serviços ao endereço VIP do *cluster* de Zabbix:

```
$ pcs constraint colocation add zabbix-server cluster_vip
$ pcs constraint colocation add httpd cluster_vip
```

15. Ordenar a prioridade de inicialização de um dado serviço:

```
$ pcs constraint order cluster_vip then Zabbix-server
$ pcs constraint order cluster_vip then httpd
```

16. Definir de forma estática, qual dos dois servidores do *cluster* é o ativo ou o passivo. Neste caso foi definido o servidor zabbix1 como servidor primário:

```
$ pcs constraint location cluster_vip then zabbix1
$ pcs constraint location Zabbix-server then zabbix1
$ pcs constraint location httpd then zabbix1
```

17. Iniciar o serviço `pacemaker` e colocar o processo permanente no arranque do sistema:

```
$ systemctl start pacemaker
$ systemctl enable pacemaker
```

18. Testar o servidor Zabbix que está definido como primário e os serviços que estão associados ao Pacemaker:

```
[root@zabbix1 ~]# sudo pcs status
cluster name: zabbixserver
Stack: corosync
Current DC: zabbix1 (version 1.1.20-5.el7_7.2-3c4c782f70) - partition with quorum
Last updated: Sun Jun 28 21:31:53 2020
Last change: Thu Jun 11 15:15:08 2020 by hacluster via crmd on zabbix2

2 nodes configured
4 resources configured

Online: [ zabbix1 zabbix2 ]
Full list of resources:
 cluster_vip    (ocf::heartbeat:IPaddr2):       Started zabbix1
 Resource Group: grp_zabbix_httpd
 zabbix_server (systemd:zabbix-server): Started zabbix1
 httpd         (systemd:httpd):         Started zabbix1
```

Anexo XXI. Configuração de monitorização distribuída com Zabbix-*proxy*

1 Instalação e configuração do Zabbix-*proxy* SQLite 3

Foram seguidos os seguintes passos genéricos para instalar o Zabbix-*proxy* em ambos os sistemas informáticos que decompõe o *cluster* de *proxys*:

1. Instalação do repositório `zabbix-proxy-sqlite3` (versão: 4.4.3-1.el7) em Centos 7:

```
# proxy1 e proxy2
$ sudo rpm -Uvh https://repo.zabbix.com/zabbix/4.4/rhel/7/x86_64/zabbix-proxy-sqlite3-
4.4.3-1.el7.x86_64.rpm
$ yum clean all
```

2. Instalação de dependências da aplicação:

```
# proxy1 e proxy2
$ sudo yum install sqlite3 sqlite-devel -y
```

3. Para guardar os dados de armazenamento no ficheiro SQLite, é necessário criar uma diretoria e dar as respetivas permissões:

```
# proxy1 e proxy2
$ sudo mkdir /var/lib/sqlite/
$ sudo chown zabbix:zabbix -R /var/lib/sqlite
$ sudo chmod 777 /var/lib/sqlite/ -R
```

4. Importar o esquema de dados e tabelas do `zabbix-proxy-sqlite3`, para a criação do ficheiro "`zabbix_proxy.db`" com a respetiva base de dados:

```
# proxy1 e proxy2
$ sudo zcat /usr/share/doc/zabbix-proxy-sqlite3-4.4.3/schema.sql | sqlite3
/var/lib/sqlite/Zabbix_proxy.db
```

5. Editar o ficheiro de configuração do Zabbix-*proxy* e associar ao parâmetro "`DBName`", o caminho do ficheiro da base de dados instanciada no ponto 4:

```
# proxy1 e proxy2
$ sudo nano /etc/zabbix/zabbix_proxy.conf

### Option: DBName
# Database name.
# For SQLite3 path to database file must be provided. DBUser and DBPassword are ignored.
# Warning: do not attempt to use the same database Zabbix server is using.
# Mandatory: yes
# Default:´

DBName=/var/lib/sqlite/zabbix.db
```

2 Associação da aplicação Zabbix-*proxy* ao *cluster* de *proxys* (Pacemaker)

1. Desativar o serviço `zabbix-proxy` nos dois servidores de *proxy*:

```
#proxy1 e proxy2
$ systemctl disable zabbix-proxy
```

2. Possibilitar a gestão processo `zabbix proxy` através do Pacemaker (configurar apenas num dos servidores):

```
$ pcs resource create zabbix-proxy systemd: Zabbix-proxy op monitor interval=3s
```

3. Associar o serviço aplicacional ao endereço VIP do *cluster* de *proxys*:

```
$ pcs constraint colocation add zabbix-proxy cluster_vip
```

4. Ordenar a prioridade de inicialização de um dado serviço:

```
$ pcs constraint order cluster_vip then Zabbix-proxy
```

5. Definir de forma estática, qual dos dois servidores do *cluster* é o ativo ou o passivo. Neste caso foi definido o servidor proxy1 como servidor primário:

```
$ pcs constraint location Zabbix-proxy then proxy1
```

6. Reiniciar o serviço `pacemaker`:

```
$ systemctl restart pacemaker
```

3 Parametrização do funcionamento em modo ativo de monitorização

O modo de funcionamento ativo do Zabbix-*proxy* permite uma monitorização independente do servidor Zabbix. As configurações realizadas descrevem o procedimento realizado na configuração (Figura 97 e Figura 98), e foram seguidas com base nos seguintes passos fundamentais:

- Definir o modo de monitorização do Zabbix-*proxy*;
- Atribuir o endereço IP do servidor Zabbix, sendo que neste caso é o endereço VIP do *cluster* de monitorização;
- Determinar uma porta TCP/IP. Por omissão a porta de comunicação entre o Zabbix-*proxy* e o Zabbix-Server é a porta TCP/IP 10051;
- Parametrizar o nome do Zabbix-*proxy*. Deve ser igual ao nome definido do lado do Zabbix-Server (*frontend*);
- O endereço de origem, o qual é utilizado na comunicação para monitorização de *hosts* via Zabbix-*proxy*, deve ser substituído para o endereço VIP do *cluster* de *proxys*. Deste modo, os *hosts* que estão a ser monitorizados não conhecem o endereço do servidor Zabbix.

```
# proxy1 e proxy2
$ sudo nano /etc/zabbix/zabbix_proxy.conf
############ GENERAL PARAMETERS #################

### Option: proxyMode
#       proxy operating mode.
#       0 - proxy in the active mode
#       1 - proxy in the passive mode
# Mandatory: no
proxyMode=0
### Option: Server
#       If proxyMode is set to active mode:
# Mandatory: yes
# Default:
Server=10.200.100.113

### Option: ServerPort
#       Port of Zabbix trapper on Zabbix server.
#       For a proxy in the passive mode this parameter will be ignored.
# Range: 1024-32767
# Default:
ServerPort=10051
```

Figura 97 - Configuração do Zabbix-*proxy* em modo ativo 1/2

```
### Option: Hostname
#       Unique, case sensitive proxy name. Make sure the proxy name is known to the server!
#       Value is acquired from HostnameItem if undefined.
#
# Mandatory: no
# Default:
Hostname=zabbix-proxy-active

### Option: SourceIP
#       Source IP address for outgoing connections.
#
# Mandatory: no
# Default:
SourceIP=10.200.100.123

### Option: DBHost
#       Database host name.
#       If set to localhost, socket is used for MySQL.
#       If set to empty string, socket is used for PostgreSQL.
# Mandatory: no
# Default:
DBHost=zabbix_proxy1
```

Figura 98 - Configuração do Zabbix-*proxy* em modo ativo 2/2

3.1 Configuração do método de armazenamento de dados

Por definição o Zabbix-*proxy* guarda os dados provenientes dos sistemas que monitoriza durante uma hora, após se verificar um problema de disponibilidade por parte do servidor Zabbix. Este valor que controla a periodicidade com que os dados são guardados é definido no parâmetro `proxyOfflineBuffer` e quantificado em unidades horárias. Na configuração que se segue é representado o processo de armazenamento para duas horas.

```
# proxy1 e proxy2
$ sudo nano /etc/zabbix/zabbix_proxy.conf


### Option: proxyOfflineBuffer
#       proxy will keep data for N hours in case if no connectivity with Zabbix Server.
#       Older data will be lost.
#
# Mandatory: no
# Range: 1-720
# Default:

proxyOfflineBuffer=2
```

4 Otimização do desempenho do Zabbix-*proxy*

```
# proxy1 e proxy2
$ cat /etc/zabbix/zabbix_proxy.conf | grep ^[^#]

######### PROXY SPECIFIC PARAMETERS #############
ConfigFrequency=20
Timeout=20
############ ADVANCED PARAMETERS ################
StartPollers=20
StartPollersUnreachable=10
StartTrappers=10
StartPingers=5
StartHTTPPollers=5
CacheSize=1G
HistoryCacheSize=1G
LogSlowQueries=3000
```

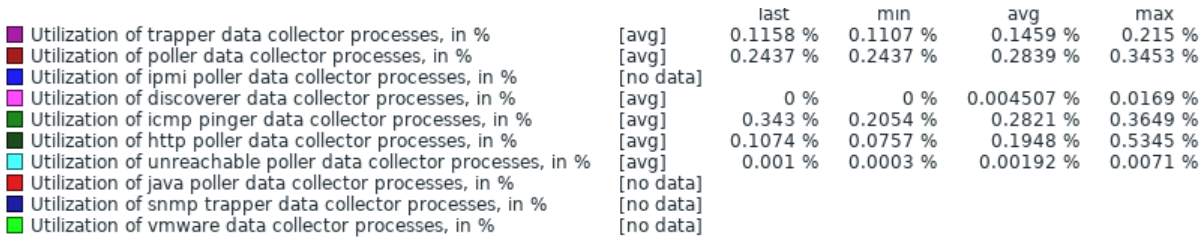


Figura 99 - Utilização dos processos internos do Zabbix-*proxy*

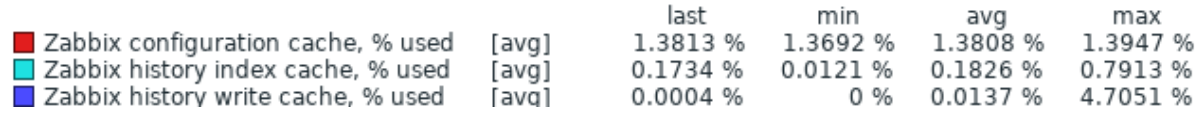


Figura 100 - Memoria cache atual do Zabbix-*proxy*

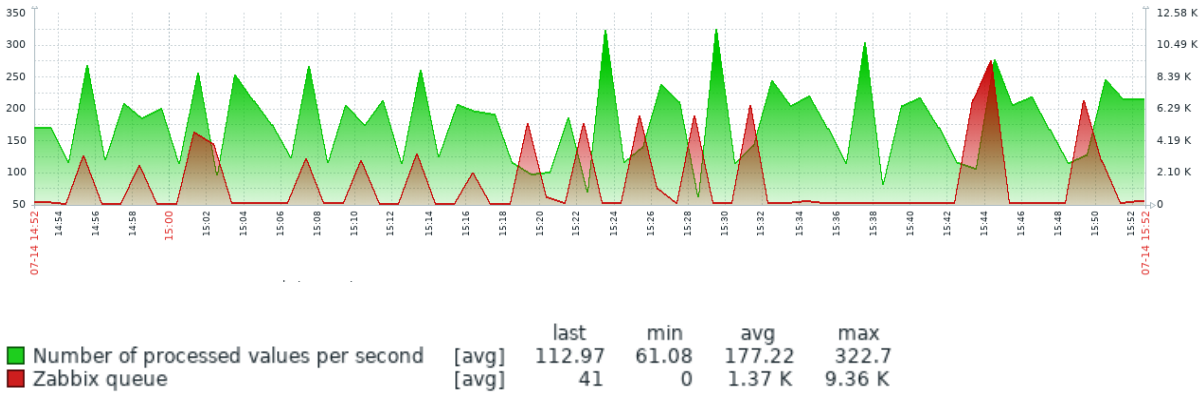
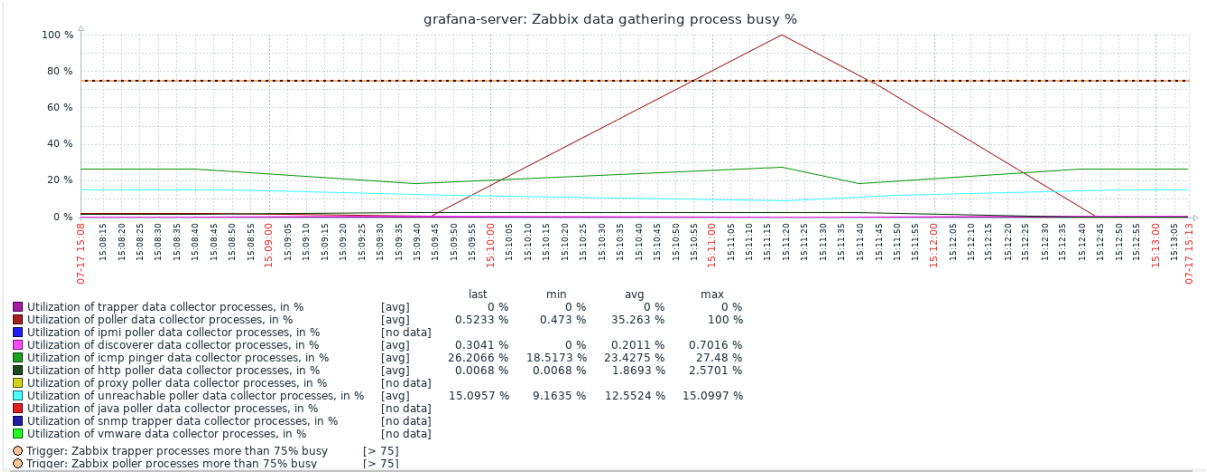


Figura 101 - Gráfico de desempenho global do Zabbix-*prox*

Anexo XXII. Otimização de desempenho dos servidores Zabbix

```
# zabbix1 e zabbix2
$ cat /etc/zabbix/zabbix_proxy.conf | grep ^[^#]

############ ADVANCED PARAMETERS #################
StartPollers=100
StartPollersUnreachable=100
StartTrappers=80
StartPingers=50
StartHTTPPollers=15
MaxHousekeeperDelete=500000
CacheSize=2G
CacheUpdateFrequency=1800
StartDBSyncers=30
HistoryCacheSize=512M
HistoryIndexCacheSize=256M
TrendCacheSize=512M
ValueCacheSize=100M
Timeout=20
LogSlowQueries=10000
```



Anexo XXIII. Configuração de agente Zabbix em sistemas computacionais

Neste anexo apresentam-se os passos de configuração dos agentes Zabbix para a monitorização de sistemas computacionais.

Todos os sistemas que não estão no mesmo domínio *broadcast* dos servidores da solução de monitorização, são monitorizados com auxílio ao *cluster* de Zabbix-*proxy*. Por sua vez, a monitorização dos componentes de monitorização (*cluster* de base de dados, *proxys* e *frontend* Zabbix) são monitorizados pelo próprio *cluster* Zabbix.

A título de exemplo, a parametrização que se segue cobre os dois cenários estudados e referenciados na secção 5.7.4, nomeadamente: o modo de funcionamento dos agentes Zabbix (passivo e ativo), bem como, as configurações adicionais para uma topologia de monitorização com ou sem recurso a *proxys* proprietários.

Enumeram-se os seguintes passos genéricos para instalar o `zabbix-agent` em sistemas informáticos:

1. Instalação do repositório `zabbix-agent` (versão: 4.4.3-1.el7) em Centos 7:

```
# servidores a monitorizar
$ sudo rpm -ivh https://repo.zabbix.com/zabbix/4.4/rhel/7/x86_64/zabbix-agent-4.4.3-
1.el7.x86_64.rpm
$ yum clean all
```

2. Instalação de dependências da aplicação:

```
# proxy1 e proxy2
$ sudo yum install zabbix-agent -y
```

3. Editar o ficheiro de configuração do `zabbix-agent` para parametrizar o modo passivo de funcionamento, utilizando o endereço IP de destino do servidor, ao qual, este se irá ligar para facultar os dados de monitorização local (Zabbix-*proxy* ou servidor Zabbix):

```
# exemplo de de configuração para zabbix-agent em modo passivo
$ sudo nano /etc/zabbix/zabbix_agentd.conf
```

Figura 102 - Configuração do agente Zabbix em modo passivo 1/2

Salienta-se que o endereço IP de destino do servidor, ao qual este agente se irá ligar para facultar os dados de monitorização, pode ser respetivo ao Zabbix-*proxy* ou ao servidor Zabbix, conforme representa a Figura 103.

```
##### Passive checks related
### Option: Server
#       List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix
servers and Zabbix proxies.
#       Incoming connections will be accepted only from the hosts listed here.
#       and '::/0' will allow any IPv4 or IPv6 address.
#       '0.0.0.0/0' can be used to allow any IPv4 address.
#       Example: Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com
# Mandatory: yes, if StartAgents is not explicitly set to 0
# Default:

Server=10.200.100.123 # → endereço IP do zabbix-proxy
#ou
Server=10.200.100.111,10.200.100.113 # → endereço IP/VIP do zabbix-server1
```

Figura 103 - Configuração do agente Zabbix em modo passivo 2/2

É ainda pertinente referir, que os servidores que recorrem a endereços IP virtuais para fazerem resiliência podem responder a pedidos com dois endereços distintos mediante a eleição da aplicação que controla o *cluster*. Nesse sentido, identificou-se que devem ser colocados os dois endereços IP que o servidor pode ter (associar IPs ao campo `Server`).

4. Iniciar o serviço `zabbix-agent` e colocar o processo permanente no arranque do sistema:

```
$ systemctl start zabbix-agent
$ systemctl enable zabbix-agent
```

Caso seja pretendido a configuração do agente Zabbix em modo ativo, as alterações são bastantes simples e passivas de serem aplicadas, tendo em conta dois fatores importantes:

- O nome do servidor deve ser definido no campo `Hostname` do agente, tendo este de ser igual ao nome instanciado no ambiente gráfico do servidor Zabbix (campo `Host name`);
- Substituir e redefinir os endereços IP no campo `ServerActive`;

```
##### Active checks related
### Option: Hostname
#       Unique, case sensitive hostname.
#       Required for active checks and must match hostname as configured on the server.
#       Value is acquired from HostnameItem if undefined.
# Mandatory: no
Hostname=Zabbix-server1
```

Figura 104 - Configuração do agente Zabbix em modo ativo 1/2

```
##### Active checks related
### Option: ServerActive
#       List of comma delimited IP:port (or DNS name:port) pairs of Zabbix servers and Zabbix
proxies for active checks.
#       If port is not specified, default port is used.
#       IPv6 addresses must be enclosed in square brackets if port for that host is specified.
#       If port is not specified, square brackets for IPv6 addresses are optional.
#       If this parameter is not specified, active checks are disabled.
#       Example: ServerActive=127.0.0.1:20051,zabbix.domain,[::1]:30051,::1,[12fc::1]
# Mandatory: no
# Default:
ServerActive=10.200.100.111,10.200.100.113 # → endereço IP/VIP do zabbix-server1
```

Figura 105 - Configuração do agente Zabbix em modo ativo 2/2

Anexo XXIV. Configuração de MIBs proprietárias no sistema de monitorização Zabbix

Para a retrocompatibilidade da solução de monitorização desenvolvida, integrou-se as MIBs disponibilizadas pelos fabricantes dos dispositivos que compõem a rede de dados da U. Porto. No caso em particular, tanto o *cluster* de *proxys* quanto o *cluster* Zabbix foram parametrizados de modo a traduzir os OIDs baseados nas MIBs adicionadas.

Descreve-se de seguida os passos realizados para inclusão das MIBs nos sistemas computacionais referidos, bem como os testes que devem ser realizados para emular a tradução de "nomes" nos respetivos OIDs e vice-versa.

1. Localizar a diretoria predefinida que disponibiliza as MIBs do sistema:

```
# zabbix-proxys e zabbix servers
$ net-snmp-config --default-mibdirs
/root/.snmp/mibs:/usr/share/snmp/mibs
```

2. Criar um diretório para guardar as novas MIBs:

```
# zabbix-proxys e zabbix servers
$ mkdir -p /usr/local/share/snmp/mibs
```

3. Editar o ficheiro `snmp.conf` e modificar o caminho da diretoria *default* para a diretoria criada no passo dois:

```
# zabbix-proxys e zabbix servers
$ vi /etc/snmp/snmp.conf
# mibdirs +/usr/share/snmp/mibs
mibdirs +/usr/local/share/snmp/mibs
```

4. Fazer o *download* da MIB proprietária (exemplo: `MIB-ENTITY` Cisco) para um diretório temporário, copiando para a diretoria criada no passo dois:

```
# zabbix-proxys e zabbix servers
$ wget ftp://ftp.cisco.com/pub/mibs/v2/ENTITY-MIB.my -P /tmp
$ cp /tmp/ENTITY-MIB.my /usr/local/share/snmp/mibs
```

5. Reiniciar os seguintes serviços nos *clusters* de Zabbix-*proxy* e nos *clusters* Zabbix:

```
$ service zabbix-server restart
$ service zabbix-proxy restart
$ service pacemaker restart
$ service snmpd restart
```

6. Demonstração das MIBs que foram integradas nos sistemas computacionais:

```
[root@zabbix_server1 mibs]# ls -la
total 3196
drwxr-xr-x 2 root root     323 Feb 28 15:58 .
drwxr-xr-x 3 root root      32 Jan 25  2020 ..
-rw-r--r-- 1 root root    1567 Feb 23  2020 AIRESPACE-REF-MIB.my
-rw-r--r-- 1 root root   49384 Jan 25  2020 CISCO-FIREWALL-MIB.my
-rw-r--r-- 1 root root   76731 Feb 28 15:55 CISCO-L4L7MODULE-RESOURCE-LIMIT-MIB.my
-rw-r--r-- 1 root root   16214 Jan 25  2020 CISCO-MEMORY-POOL-MIB.my
-rw-r--r-- 1 root root  104190 Jan 25  2020 CISCO-PROCESS-MIB.my
-rw-r--r-- 1 root root   63658 Jan 25  2020 CISCO-QOS-PIB-MIB.my
-rw-r--r-- 1 root root   72246 Jan 25  2020 CISCO-REMOTE-ACCESS-MONITOR-MIB.my
-rw-r--r-- 1 root root   16811 Jan 25  2020 CISCO-SMI.my
-rw-r--r-- 1 root root  102922 Jan 25  2020 CISCO-TC.my
-rw-r--r-- 1 root root   59499 Jan 25  2020 ENTITY-MIB.my
-rw-r--r-- 1 root root 2683954 Feb 19  2020 PowerNet-MIB.mib
```

7. Testar a tradução de nomes de serviços a monitorizar em OIDs:

```
# snmptranslate -IR -On <mib_name>::<name or OID_Text>
$ snmptranslate -IR -On CISCO-PROCESS-MIB::cpmCPUTotal5minRev
.1.3.6.1.4.1.9.9.109.1.1.1.1.8
```

Anexo XXV. Descoberta de baixo nível de *hosts* e serviços

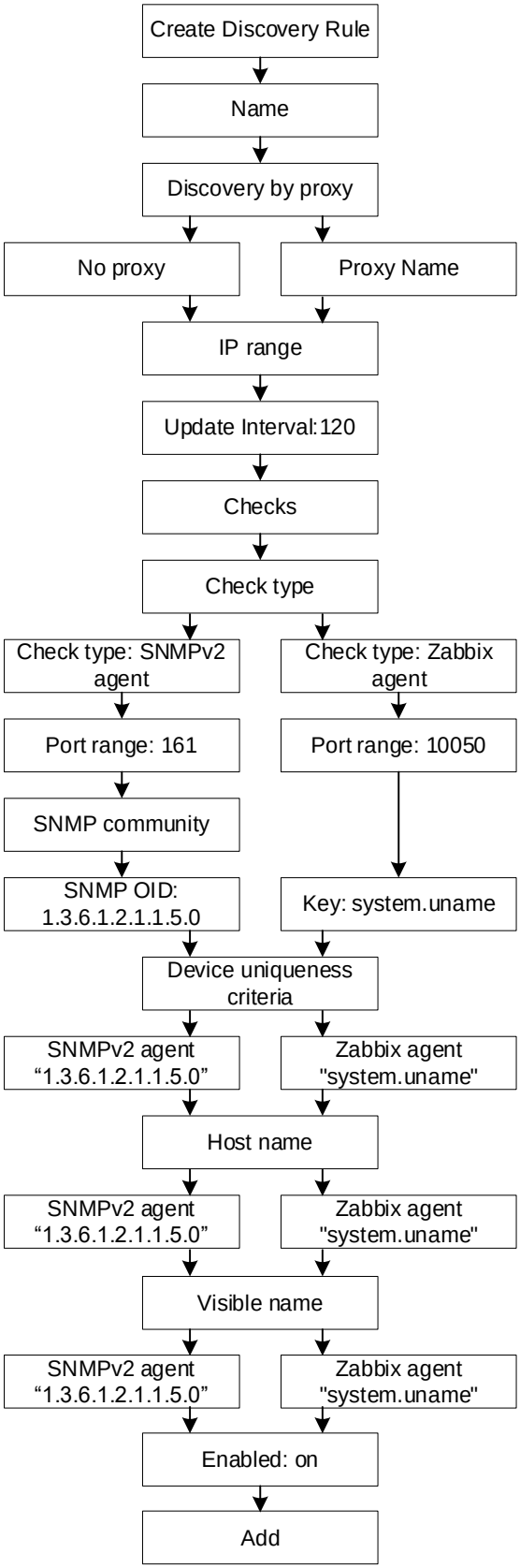


Figura 106 - Diagrama de configuração de protocolos para suporte da LLD

Estre anexo retrata um exemplo de configuração para a descoberta automática de equipamentos (LLD), baseando-se no sistema de monitorização Zabbix.

O diagrama de blocos da Figura 106 representa a modelo de configuração para dois exemplos de agentes (SNMP e Zabbix-*agent*), de modo a proceder ao processo automatizado da monitorização de rede.

Para que este procedimento seja finalizado, devem ser parametrizadas as ações pretendidas após a regra de descoberta ser finalizada. No diagrama da Figura 107 é dado o exemplo para validar os equipamentos de rede e sistemas computacionais mediante as condições que pretendemos configurar. Depois desse processo, as operações permitem adicionar os *hosts* a diferentes grupos, associando também os *templates* de monitorização de forma dinâmica, bem como permite ainda adicionar os sistemas ao inventário, entre outras opções.

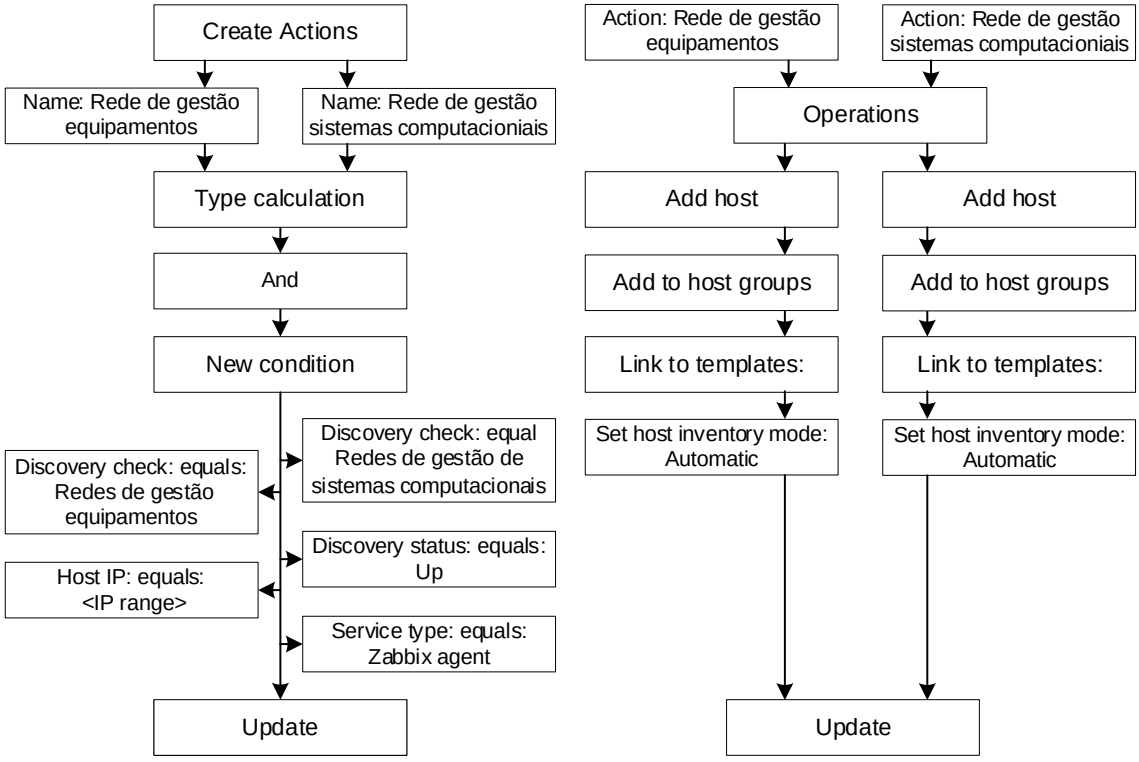


Figura 107 - Diagrama para parametrização de ações/operações de monitorização dinâmica

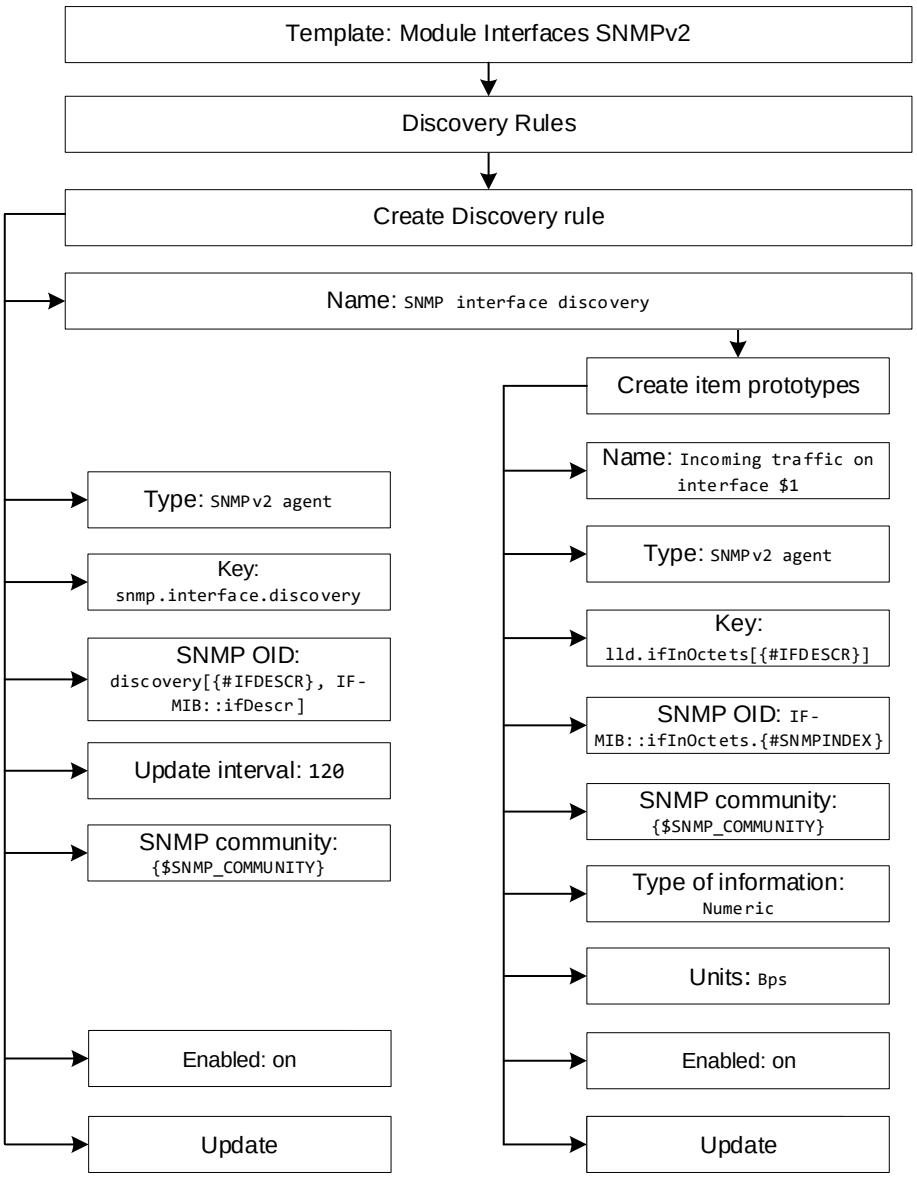


Figura 108 - Diagrama de configuração para descoberta de serviços SNMP LLD

No que concerne ao domínio conceptual do modelo arquitetónico para a descoberta de serviços SNMP LLD (Figura 108), destaca-se o campo mais importante, sendo este o parâmetro SNMP OID. Esta expressão define-se pelo seguinte formato:

SNMP OID: `discovery[{#MACRO1}, oid1, {#MACRO2}, oid2, …,]`

Cada macro (`{#MACRO1}` e `{#MACRO2}`) determina um nome válido para a LLD, por sua vez cada OID (`oid1` e `oid2`) gera um valor com significado para a respetiva macro.

Para além das macros estritamente especificadas, o Zabbix necessita de um protótipo de item, no qual deve ser estabelecido a macro (`{#SNMPINDEX}`) que contem a extensão de OID de cada entidade encontrada para a raiz da MIB em particular (`IF-MIB::ifDescr`).

```
$ snmpwalk -v2c -c <SNMP_COMMUNITY> <IP host> IF-MIB::ifDescr
IF-MIB::ifDescr.2 = STRING: MgmtEth0/RSP0/CPU0/0
IF-MIB::ifDescr.3 = STRING: MgmtEth0/RSP0/CPU0/1
IF-MIB::ifDescr.4 = STRING: TenGigE0/3/0/0
IF-MIB::ifDescr.5 = STRING: TenGigE0/3/0/1
IF-MIB::ifDescr.6 = STRING: GigabitEthernet0/0/0/0
IF-MIB::ifDescr.7 = STRING: GigabitEthernet0/0/0/1
IF-MIB::ifDescr …
```

Figura 109 - Exemplo de SNMP index respetivos à MIB::ifDescr

Contrariamente ao processo de descoberta de serviços com a utilização do SNMP LLD, o modelo de descoberta para sistemas computacionais baseado em agente Zabbix, recorre ao campo “KEY”.

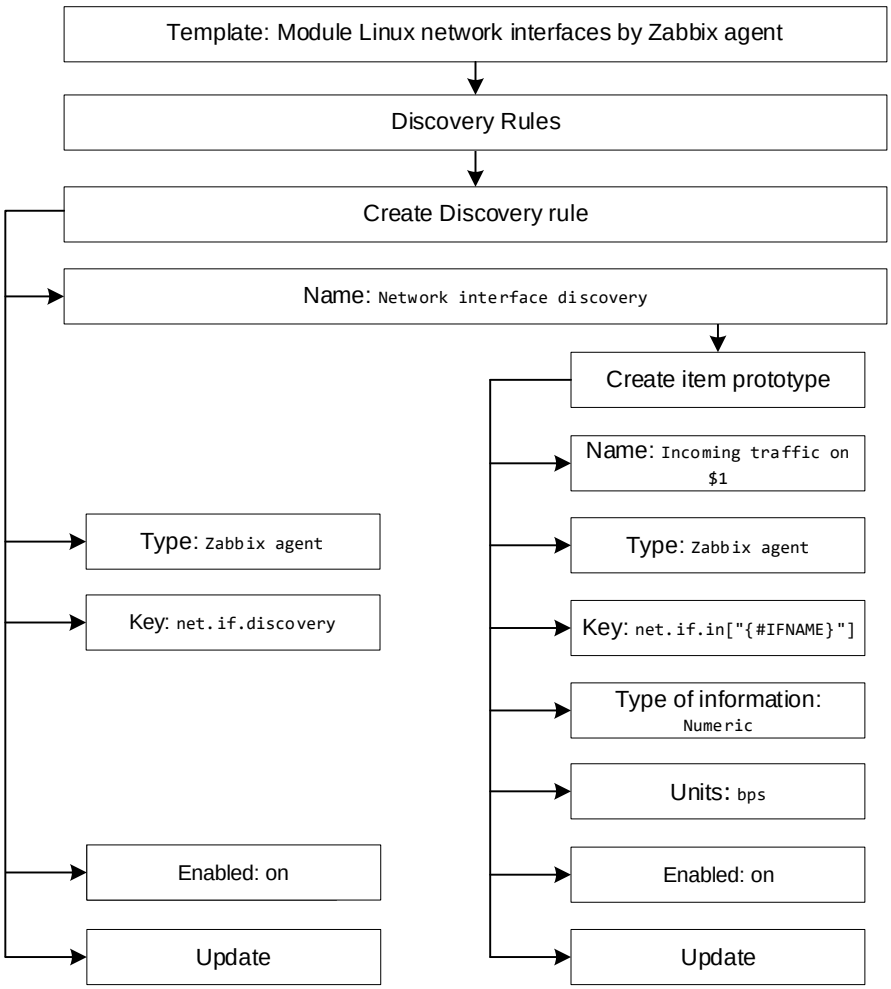


Figura 110 - Diagrama de configuração para descoberta de serviços via agente Zabbix

Este parâmetro ganha um enorme relevo, utilizando expressões que invocam *scripts* genéricos (ou parametrizáveis) com uma sintaxe sustentada no formato JSON.