SISTEMA DE CORRELAÇÃO DE EVENTOS E NOTIFICAÇÕES - SCEN

Carlos Alexandre Correia Leite da Silva

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto
2010

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: Carlos Alexandre Correia Leite da Silva, Nº 1920550, 1920550@isep.ipp.pt
Orientação científica: Prof. Doutor Jorge Botelho Costa Mamede, jbm@isep.ipp.pt
Co-orientação científica: Eng. Hélder Vieira Mendes, hvm@isep.ipp.pt

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Telecomunicações
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto
6 de Dezembro de 2010

Agradecimentos

Gostaria de aproveitar este espaço para agradecer a todos aqueles que, de alguma forma directa ou indirecta, contribuíram para o desenvolvimento deste trabalho. Que foi a nível de conhecimento, uma experiência bastante motivadora e muito enriquecedora. Ao Prof. Doutor Jorge Botelho Costa Mamede meu Orientador científico e ao Eng. Hélder Vieira Mendes meu co-orientador e supervisor, por todo o apoio científico e pedagógico e pelo rumo sustentado e traçado desde o inicio. À minha família agradeço o apoio sempre manifestado. A minha esposa, Sónia Leite da Silva, pela sua permanente compreensão e forte motivação, em todos os momentos desta jornada. Para os meus dois filhos Bernardo e Mafalda uma palavra de desculpa pela minha ausência em vários momentos das suas vidas. À minha mãe, sempre com uma palavra amiga, muito envolvida nas etapas que percorri e sempre muito orgulhosa em tudo o que faço. Por último, a agradeço a todos os meus amigos, que para não me esquecer de ninguém, ficarão no anonimato.

Resumo

As redes actuais, com um crescimento de eventos nos sistemas de gestão cada vez mais sofisticado, um dos grandes ojectivos para o futuro é condensarem uma grande quantidade desses eventos num pequeno número de eventos, mas mais significativo para o relatório de falhas. Esta necessidade prática pode ser realizada recorrendo a mecanismos de correlação de eventos. A correlação de eventos é uma área de intensa investigação na comunidade científica e industrial. Neste contexto surgiu o desejo de realizar um projecto no âmbito da correlação de eventos em redes com gestão e monitorização na infra-estrutura SNMP. O objectivo do trabalho realizado foi a definição e criação de uma infra-estrutura de software com capacidade de correlacionar autonomamente e de forma inteligente os eventos recebidos a partir de sistemas informáticos.

A ideia para a criação desta infra-estrutura nasce da dificuldade que surge na constante configuração e adaptação dos parâmetros de análise dos actuais sistemas. É por isso, apresentado uma plataforma de valor acrescentado que permite auxiliar os responsáveis pela gestão de infra-estruturas de sistemas informáticos. Para melhorar a sua eficácia na resolução dos problemas dos seus sistemas e redes, através da observação de um subconjunto de eventos que é relevante na globalidade de eventos recebidos. Para tal foram estudadas outras soluções, várias ferramentas comerciais, de fonte de código aberto e foram estudadas as suas características e os seus modelos. Foi posteriormente desenvolvido um novo modelo adaptado com base numa ferramenta escolhida e que contempla a interacção de vários elementos de monitorização de rede.

Estes elementos criam a capacidade de observar mudanças de estado dos serviços definidos e corrrelacionar as referidas alterações com os eventos que vão sendo obtidos dos sistemas. A infra-estrutura de monitorização proposta visa assim permitir a avaliação da relevância dos eventos recebidos a partir de sistemas de pooling ou de notificação e daí inferir a importância dos eventos, deixando de ser necessário ao administrador da plataforma de gestão ou ao administrador de sistema ter esse trabalho. Para validar o modelo foi implementado um laboratório virtual onde foram criados os elementos constituintes do modelo proposto e foram feitas simulações, com vista à obtenção e validação de

resultados. Como conclusão, a infra-estrutura que foi definida e testada reflectiu o funcionamento pretendido, baseando-se apenas nas definições preexistentes sobre os serviços monitorizados, no conhecimento das bases de dados do sistema de monitorização existente através de pooling e cruzamento de tabelas das bases de dados, e dos vários tipos de equipamentos de rede e nas suas mensagens de estado. No final foram apresentados a resposta do sistema á saída e desenvolvimento futuro.

Palavras-Chave

Gestão e monitorização de redes, SNMP, ferramentas Open Source, SCEN, NMSIS, correlação, SEC, raiz da causa do problema, inferido, detecção, amostragem padrão, correspondência, algoritmo de correlação, causalidade, correlação temporal, correlação causal, raciocínio, evento, acção, serviço, servidor, comutador, encaminhador.

Abstract

Current networks, with increasingly sophisticated systems management events, one of the major objectives is to condense a large amount of these events in a small number, but more significative in the report of their failures. This practical necessity can be accomplished through event correlation mechanisms. Event correlation is an area of intense research in scientific and industrial community. From this context, came the desire to accomplish a project work within the network event correlation data management based on SNMP infrastructure.The aim of this work was the development of an infrastructure for monitoring with the ability to correlate autonomously and intelligently events received from computer systems.

The idea for the creation of this infrastructure was born of the difficulty that arises on the constant setting and adjustment of parameters of analysis of existing systems. It is presented as a value-added platform, that enables help who those responsible for managing an infrastructure of computer systems. To improve the effectiveness of the resolution of the problems of their systems and networks, through observation of a subset of events that is relevant to the whole of events received. For such other solutions were studied, several commercial tools and open source code, were studied their characteristics and their models. Was subsequently developed a new model adapted on the basis of the chosen tool and that contemplates the interaction of various elements of network monitoring.

These elements that create the ability of observe state changes and correlation services defined these amendments with the events that are being obtained from the systems. Infrastructure monitoring proposal aims therefore allow an assessment of the relevance of the events received from pool systems or notification and inferred the importance of events, leaving to be necessary for the management platform administrator or system administrator to have this job. To validate the model was implemented a virtual lab where they were created the constituent elements of the proposed model simulations, and were made with a view to obtaining and validating results. As a conclusion, the infrastructure that was defined and tested reflected the intended operation, relying only on pre-existing settings on services monitored, knowledge of databases existing monitoring system

through pooling and crossing connecting tables of databases, and various types of network equipment and its status messages. At the end were presented the system response will output and future development.

Keywords

Monitoring and management networks, SNMP, Open Source tools, SCEN, NMSIS, correlation, SEC, problem root cause, inferred, detection, sampling pattern, matching, correlation algorithm, causality, temporal correlation, causal correlation, reasoning, event, action, service, server, switch, router (gateway).

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

AJAX	–	JavaScript and XML
API	–	Application Programming Interface
ARP	–	Address Resolution Protocol
ARPA	–	Advanced Researche Project Agency
ASCII	–	American Standard Code for Information Interchange
ASN.1	–	Abstract Syntax Notation One
ATM	–	Asynchronous Transfer Mode
BASE	–	Basic Analysis and Security Engine
BIOS	–	Basic Input/Output System
BSD	–	Berkeley Software Distribution
BSD	–	Berkeley Software Distribution
CBR	–	Case-based reasoning
CC	–	Codebook correlation
CD	–	Compact Disk
CDP	–	Cisco Discovery Protocol
CERT	–	Computer Emergency Response Team
CGI	–	Common Gateway Interface
CIM	–	Common Information Model

CLI	–	Command Line Interface
CMIP	–	Common Management Information Protocol
CMU	–	Carnegie Mellon University
COPS-PR	–	Common Open Policy Service for Provisioning
CORBA	–	Common Object Request Broquer Arquitecture
CPU	–	Communications Processor Unit
DBMS	–	DataBase Management System
DBSM	–	DataBase Synchronization Module
DEG	–	Departamento de Engenharia Electrotécnica
DEG	–	Departamento de Engenharia Geotécnica
DHCP	–	Dynamic Host Configuration Protocol
DiffServ	–	Differentiated Services
DMI	–	Desktop Management Interface
DNS	–	Domain Name System
DTMF	–	Desktop Management Task Force
EGP	–	Exterior Gateway Protocol
EIGRP	–	Enhanced Interior Gateway Routing Protocol
FAT	–	File Allocation Table
FCAPS	–	Fault, Configuration, Accounting, Performance, Security
FIFO	–	First In First Out
FMP	–	Fault Management Platform

FOG	–	Free Open Source Ghost
FSF	–	Free Software Foundation
FTP	–	File Transfer Protocol
FW	–	Firewall
GESM	–	Group Equipment Synchronization Module
GNU	–	General Public License
GRED	–	Generic Random Early Drop
GUID	–	Globaly Unicode Identifier
HFS	–	Hierarchical FileSystem
HFSC	–	Hierarchical Fair Service Curve
HP	–	Hewlett Packard
HPFS	–	High Performance File System
HTB	–	Hierarchical Token Bucket
HTC	–	High Tech Computer Corporation
HTTP	–	Hypertext Transfer Protocol
HTTPD	–	Hypertext Transfer Protocol Daemon
HTTPS	–	Hypertext Transfer Protocol over Secure Socket Layer
IBM	–	International Business Machines Corporation
ICMP	–	Internet Control Message Protocol
IDE	–	Integrated Drive Electronics
IDM	–	Identity Drives Manter

IETF	–	Internet Engineering Task Force
IGRP	–	Interior Gateway Routing Protocol
IntServ	–	Integrated Services
IP	–	Internet Protocol
IPP	–	Instituto Politécnico do Porto
ISEP	–	Instituto Superior de Engenharia do Porto
ISO	–	International Organization of Standardization
ISODE	–	ISO Development Environment
ITU	–	International Telecommunication Union
LAN	–	Local Area Network
LDAP	–	Lightweight Directory Access Protocol
LLDP	–	Link Layer Discovery Protocol
LVM	–	Logical Volume Management
MAC	–	Media Access Control
MAN	–	Metropolitan Area Network
MBR	–	Master Boot Record
MBR	–	Model-based reasoning
MIB	–	Management Information Base
MIBv1	–	Management Information Base version 1
MIBv2	–	Management Information Base version 2
MIBv3	–	Management Information Base version 3

MIT	–	Massachusetts Institute of Technology
MRTG	–	Multi Router Traffic Grapher
MTA	–	Mail Transport Agent
MUA	–	Mail User Agent
NAT	–	Network Address Translation
NEDG	–	Network Element Dependency Graph
NMSIS	–	Network Administration Visualized
NFS	–	Network File System
NIDS	–	Network Intrusion Detection System
NMS	–	Network Management System
NMSIS	–	Network Management System with Imaging Support
NNM	–	Network Node Manager
NNTP	–	Network News Transfer Protocol
NTFS	–	New Technology File System
NTNU	–	Norwegian University of Science and Technology
OEMF	–	OpenView Element Management Framework
OID	–	Object Identifier
OSI	–	Open Systems Interconnection
OSPF	–	Open Shortest Path First
OSSIM	–	Open Source Security Information Management
PC	–	Personal Computer

PCM	–	ProCurve Manager
PCMCIA	–	Personal Computer Memory Card International Association
PDA	–	Personal Data Assistent
PDF	–	Potable Document Format
PDU	–	Protocol Data Unit
PFSM	–	Probabilistic Finite State Machine
PHP	–	PHP: Hypertext Preprocessor
PRIO	–	Priority Scheduler
QoS	–	Quality of Service
RAID	–	Redundant Array of Independent Drives
RAM	–	Random Access Memory
RARP	–	Reverse Address Resolution Protocol
RAW	–	Data without filesystem
RBR	–	Rule-based reasoning
RFC	–	Request For Comments
RIP	–	Routing Information Protocol
RRD	–	Round Robin Database
SCLI	–	SNMP Command Line Interface
SEC	–	Simple Event Correlator
SH	–	Shell

SID	–	Security identifier
SIMG	–	Servidor de Imagens / Servidor FOG
SLES	–	Suse Linux Enterprise
SMB	–	Server Message Block
SMFA	–	Specific Managment Functional Areas
SMI	–	Structure of Management Information
SMIv2	–	Structure of Management Information version 2
SMS	–	Short Message Service
SMTP	–	Simple Mail Transfer Protocol
SNMSIS	–	Servidor NMSIS
SNMP	–	Simple Network Management Protocol
SNMPv1	–	Simple Network Management Protocol version 1
SNMPv2	–	Simple Network Management Protocol version 2
SNMPv3	–	Simple Network Management Protocol version 3
SPPI	–	Structure of Policy Provisioning Information
SQL	–	Structured Query Language
SSH	–	Secure Shell
SSL	–	Secure Sockets Layer
STG	–	State Transtion Graphs
SW	–	Servidor Web
TCP	–	Transmission Control Protocol

TCP/IP	–	Transmission Control Protocol / Internet Protocol
TELNET	–	Telecommunication Network
TFTP	–	Trivial File Transfer Protocol
TKIP	–	Temporal Key Integrity Protocol
TMN	–	Telecommunications Management Network
UCD	–	University of California at Davis
UDP	–	User Datagram Protocol
UFS	–	Unix FileSystem
UML	–	Unified Modeling Language
UNDI	–	Universal Network Device Interface
UUID	–	Universal Unicode Identifier
VLAN	–	Virtual LAN
WAN	–	Wide Area Network
Web	–	World Wide Web
WfM	–	Wired for Management Framework
WSDL	–	Web Services Description Language
XML	–	Extensible Markup Language

1. INTRODUÇÃO

Os avanços tecnológicos exercem hoje um grande impacto na sociedade. A informação tem-se tornado cada vez mais uma vantagem competitiva para as empresas e organizações. Cada vez mais, as empresas, para se tornarem competitivas, têm investido em tecnologia de informação, como a única forma de tornar seguro o processo de decisão. É nesse quadro que as redes de computadores proliferam, encurtando as distâncias e diminuindo o tempo de resposta entre transacções das organizações de todo o mundo. Os sistemas de computadores têm vindo a ser progressivamente interligados em rede, e as aplicações tomam partido da distribuição para oferecer mais e melhores serviços, tornando as redes e os recursos associados indispensáveis.

À medida que as redes e os sistemas distribuídos aumentam de dimensão e importância, torna-se cada vez mais necessário recorrer à gestão de redes para garantir que a rede funciona correctamente e providencia os serviços esperados pelos utilizadores. A gestão de redes tem como funções supervisionar e controlar as redes e os serviços por elas oferecidos, bem como planear modificações na rede. A elevada dimensão e complexidade das redes de dados leva a outra necessidade, a necessidade de optimizar os custos de criação, manutenção e desenvolvimentos futuros. Neste contexto a escolha por uma boa ferramenta de gestão de redes é ponto de partida para fomentar essas necessidades relativamente à manutenção das redes de dados. Neste projecto o foco principal é a adequação dos meios tecnológicos existentes na infra-estrutura de rede aos seus utilizadores ou ao seu operador de rede, de forma a facilitar o despiste de avarias que surjam na rede.

1.1. CONTEXTUALIZAÇÃO

O Departamento de Engenharia Electrotécnica (DEE) e o Departamento de Engenharia Geotécnica (DEG), tem tido um aumento substancial em tamanho e complexidade, e daí surgir a necessidade de desenvolver uma solução de software que permitisse realizar a gestão e monitorização centralizada da rede informática, de modo a garantir o serviço prestado aos seus utilizadores.

Para esse efeito surgiu o projecto [111] que desenvolveu a plataforma de trabalho *Network Management System with Imaging Support* (NMSIS). Esse trabalho procurou no seu contexto de Tese, dar resposta às dificuldades associadas ao processo de gestão. As redes actuais, pretendem cada vez mais sistemas sofisticados de gestão de eventos, que condensem uma grande quantidade desses eventos num pequeno número, mas também mais significativo.

Desta necessidade prática surgiu a ideia de desenvolver uma ferramenta de correlação de eventos para integrar no NMSIS, com o intuito de facilitar o processo de troubleshooting e despiste de avarias na rede e desconbrir de forma mais eficiente, a raiz dos problemas. Este estudo introduz uma aproximação para correlação de eventos tirados da base de dados da plataforma NMSIS e das notificações dos equipamentos activos na rede. O beneficio que este trabalho trás ao realizado anteriormente [111]**Error! Reference source not found.**, é a introdução de um novo *Sistema de Correlação de Eventos e Notificações* (SCEN).

O SCEN serve para instrumentar e dotar o sistema de gestão NMSIS com um bloco de correlação de eventos que não existia. Esta inclusão reflectiu-se no desenvolvimento de um processo de aquisição dos eventos gerados a partir do NMSIS e também na proposta de incluir no SCEN uma ferramenta de código fonte aberto, que permitisse a correlação de eventos. Esta ferramenta chamada *Simple Event Correlator* (SEC), tem como função principal a inclusão de um processo de verificação de eventos, com um motor de inferência para processamento de eventos, para o sistema reagir adequadamente de acordo com as entradas.

1.2. OBJECTIVOS

O objectivo principal deste projecto é a integração de um sistema de correlação de eventos através da aquisição de informação proveniente do sistema NMSIS. O presente trabalho

fez uma pesquisa para identificar as principais características do modelo de gestão do NMSIS e quais os requisitos para a possibilidade de ser integrado com um motor de correlação. Para o efeito foram estudadas e comparadas várias soluções no mercado *Open Source* com correlação de alarmes que atendem esses requisitos. Foram analisadas formas de como obter informação do NMSIS, que contenha certos parâmetros e variáveis de estado dos diversos elementos de rede, para formação de novos eventos. Foi testada e implementada uma solução com uma ferramenta de correlação, que depois foi integrada com o NMSIS e montada no Laboratório de Rede e Serviços de Comunicação do DEE. Foi sujeita à execução de uma bateria de testes e tiram-se resultados. A aplicabilidade da solução encontrada foi a redução substâncial no número de eventos, devido à capacidade de filtrar eventos desnecessários e condensar a informação no que é realmente importante para chegar à raíz da causa dos problemas de rede.

1.3. ORGANIZAÇÃO DO RELATÓRIO

No capítulo 1 são apresentadas as causas principais que deram origem a criação desta tese, o âmbito do projecto e os objectivos pretendidos e a organização do relatório. No capítulo 2, apresenta-se a introdução aos protocolos de gestão de redes sob o ponto de vista histórico. Foi abordado os sistemas de gestão de redes com protocolo SNMP, o seu modelo de gestão e constituição, tipos de agente, classificação da informação de gestão (MIB).

No capítulo 3 apresenta-se o estado da arte das ferramentas de gestão de alarmes com correlação de eventos. Conceito de correlação, para que serve e como pode ser utilizado. São feitas várias alusões aos tipos de correlação que pode existir entre as variáveis dos eventos (correlação causal ou temporal). São classificadas de forma genérica as operações que as ferramentas de correlação devem e podem efectuar. São também estudados os modelos de correlação com premissas que servem de base ao raciocínio, para ser possivel inferir determinado tipo de conclusões. Depois são estudadas nove ferramentas de correlação de eventos, de fonte aberta, comerciais e outras. São estudadas os seus modelos de funcionamento as sua cracteristicas genéricas e caracteristicas específias, vantagens e disvantagens. Mais tarde faz-se um estudo comparativo e por fim escolhe-se a ferramenta ideal para o sistema. Depois de escolhida a ferramenta é estudada em pormenor. Essa ferramenta de código aberto e disponível tem a designação de SEC. É estudado e apresentado o SEC, referindo as funcionalidades, o seu modelo de operação, os tipos de regras, as acções permitidas e a performance da ferramenta.

No capítulo 4, apresenta-se uma rede de teste montada no Laboratório de Rede e Serviços de Comunicação do DEE, na sala F507. Este laboratório simula uma rede real, com equipamentos activos da rede informática do DEE, como é o exemplo do sistema NMSIS e introdução de outros activos como *routers*, *switchs* e servidores de serviços. Posteriormente é descrito a constituição e modelo funcional do SCEN e apresentado os blocos constituintes: o script em PHP para fazer *querys* à base dados do NMSIS; a implementação do *daemon snmptrapd* e o *scrip*t em Perl para tratamento das *traps* originárias dos activos de rede via Net-SNMP (*daemon snmptrapd*). É feita uma a descrição do *daemon* criado neste trabalho em linguagem PHP, responsável, pela criação dos quatro tipos de querys feitas à base de dados NMSISdb. É descrito o *daemon* em linguagem Perl, criado no âmbito do trabalho e responsável pelo tratamento dos dados recebidos do Net-SNMP. Foram descritas as configurações na implementação do *snmptrapd* para processamento das *traps* via UDP e também foi também explicado a configuração nos activos de rede para execução das *traps*.

No capítulo 5, pode-se dividir em duas partes. Na primeira parte, começa-se por apresentar os eventos propriamente ditos que são processados pelo sistema. Tipos de eventos via query (são quatro tipos distintos) e os eventos do tipo via *trap*. Depois estuda-se as correlações de causalidade. Foram atribuídos símbolos às variáveis dos quatro tipos de *querys* e foi referido qual as tabelas de origem. Foi feito um quadro com as tabelas usadas e como foram cruzadas essas tabelas pelo código *daemon* concebido para o efeito neste projecto e responsável pelas *querys* feitas à base de dados. Foi criado para a rede de teste, um caso de estudo com um algoritmo de causa efeito, através de um mapa de causalidade em árvore. Depois de devidamente comprovado foram inferidas por correlação causal, quais as causas para os tipos de eventos recebidos e com o algoritmo foram criados novos eventos que filtram e condensam a informação no que é realmente importante. Paralelamente foi também elaborado para a rede de teste, um caso de estudo, com um algoritmo de correlação temporal que permite a filtrar, suprimir e generalizar eventos que podem inundar uma rede com muita informação. Para ambos foi apresentada a resposta do sistema aos estímulos dos eventos. A preocupação principal em ambos os algoritmos foi procurar uma forma de remeter a resposta do sistema para o essencial.

Na segunda parte são apresentadas e descritas de form prática as regras aplicadas e forma como as acções das regras respondem em situações reais na rede de estudo criada para o

efeito. Foram estudadas as expressões regulares em Perl que são aplicadas nas regras, os princípios funcionais das regras e as variáveis internas. Foram descritos dois tipos de regras: "*PairWithWindow"* e "*Pair*" para obter o mesmo fim. Em cada uma das regras foi descrito o seu modo *operandus* e as suas diferenças. Foi apresentado um algoritmo final de correlação causal constituído por 3 regras do tipo "*PairWithWindow"* e um algoritmo final de correlação temporal com 4 regras do tipo "*PairWithWindow",* "*Pair*", "*SingleWith2Thresholds*" e "*SingleWithSuppress*". Por fim, fez-se a exposição da resposta do sistema à saída e a forma como é apresentada as notificações do sistema à saída, tanto via mail como página WEB. Finalmente, tiram-se as principais conclusões do trabalho apresentado e as propostas para desenvolvimentos futuros.

2. SISTEMAS DE GESTÃO DE REDES

2.1. INTRODUÇÃO AOS SISTEMAS DE GESTÃO DE REDES

À medida que as redes crescem em escala e extensão, os seus recursos e aplicações, tornam-se cada vez mais indispensáveis para as organizações que as utilizam. As redes de computadores devem ser geridas com a finalidade de oferecer um serviço de qualidade aos seus utilizadores. Esta gestão envolve a monitorização dos recursos distribuídos das redes. Na sua essência, a gestão de redes procura sempre assegurar que os sistemas de informação disponíveis nas redes, estejam sempre operacionais a todo o momento.

A gestão de redes de computadores é por si só um assunto complexo. As redes tornam-se maiores (em extensão), mais complexas (na tecnologia) e heterogéneas (várias plataformas de hardware e software distintas), o que faz com que as funções de gestão sejam em si, de grande complexidade. Como a gestão não pode ser realizado somente pelo esforço humano, a complexidade da gestão de redes impõe o uso de soluções automatizadas.

Para a gestão, os equipamentos têm uma base de dados de informação de gestão, mantida por um Agente de gestão, à qual as aplicações de gestão acedem através de um protocolo

de gestão de redes [40][114][126]. Os três mais conhecidos protocolos de gestão de redes que actualmente existem com versões *standard* a nível internacional, são:

- *Common Management Information Protocol* (CMIP) da comunidade *International Organization of Standardization* (ISO);
- *Telecommunications Management Network* (TMN) da comunidade *International Telecommunication Union* (ITU);
- *Simple Network Management Protocol* (SNMP) da comunidade *Internet Engineering Task Force* (IETF) Internet Management.

Para termos uma perspectiva da evolução dos vários protocolos de gestão ao longo dos anos, foi sumarizado no próximo capitulo, a evolução de várias tecnologias e protocolos que foram desenvolvidos com esse propósito de forma a compreender quais os seus objectivos e suas principais linhas de orientação.

2.2. EVOLUÇÃO DOS PROTOCOLOS DE GESTÃO DE REDES

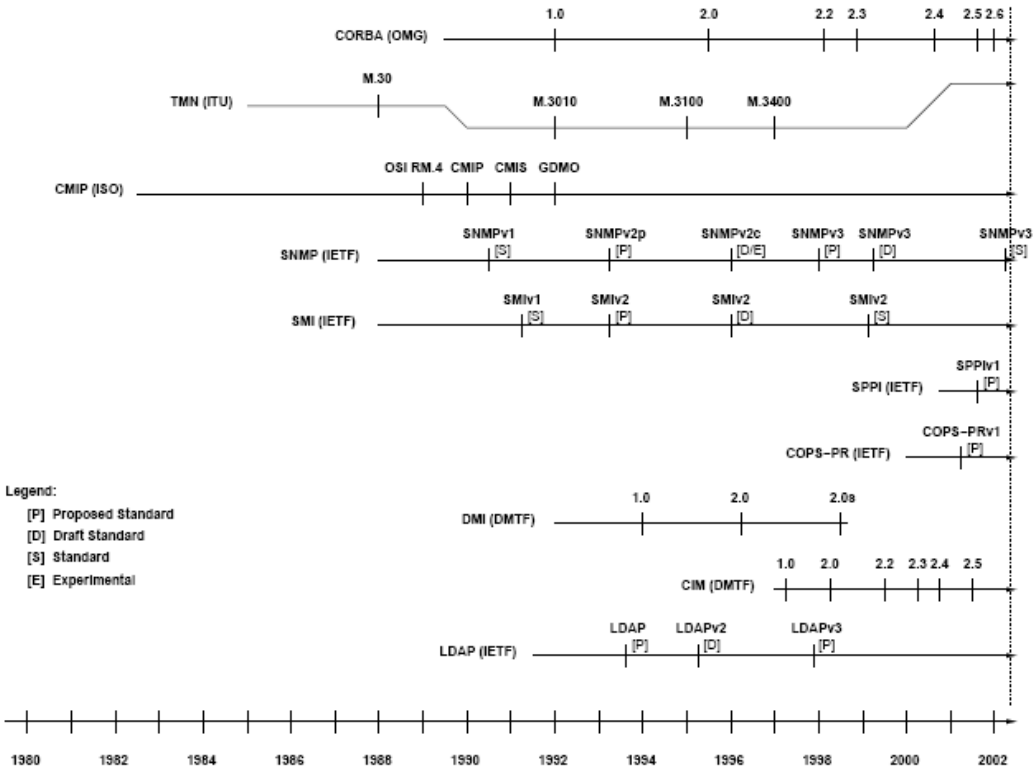


Figura 1 Perspectiva histórica das tecnologias em redes de gestão [23].

A tecnologia (SNMP teve uma grande evolução no fim dos nos 80, década de 90, nem todas as propostas de melhoramento tiveram uma história de sucesso, mas certo é, que está mundialmente disseminada nas redes actuais e é usada em larga escala para colectar estatísticas e detectar falhas de rede.

A figura 1 mostra a evolução de vários protocolos e linguagens de definição de dados que foram usados para manutenção e gestão de redes de comunicações de dados [23]. O IETF finalizou o *Simple Network Management Protocol version 1* (SNMPv1) no inicio dos anos 90 e cedo começou a reunir esforços para introduzir o SNMP versão 2 conjuntamente com a segunda versão da linguagem de definição de dados, *Structure of Management Information* (SMI). Este processo tornou-se complexo e o resultado foi a versão 2c (SNMPv2c). O SNMPv2c é ainda relativamente usado nas redes de hoje, contudo não é um produto totalmente normalizado do IETF (IETF standard), porque tem algumas lacunas ao nível de máxima segurança (que foi um dos objectivos principais do SNMPv2).

A segunda versão do SMI (SMIv2) foi mais bem sucedida e foi publicada como norma da Internet (Internet standard)[6][8]. Como o SNMPv2c não atingiu os seus principais objectivos, em 1997 iniciou-se uma tentativa de definir o SNMP versão 3 (SNMPv3)[1] que permite máxima segurança, capacidades de administração remota e provavelmente o mais importante, uma arquitectura em forma de plataforma de trabalho [9]. O SNMPv3 foi finalmente introduzido nas redes actuais, demorou cerca de uma década desde o SNMPv1 até ao SNMPv3, passando ainda por uma versão intermédia SNMPv2c. Mais detalhes em documentos que definem as várias versões de SNMP pode ser encontrada no [1]. A figura 1 mostra que o SNMP teve muitos competidores. A norma TMN definida pelo ITU para gerir redes de telecomunicações, estava até certo ponto bem alinhada nas especificações de forma muito próxima com o CMIP definido pelo ISO. Em 2002 o ITU estava mais empenhado em rumar em direcção ao *Common Object Request Broquer Arquitecture* (CORBA) como um *middleware* de comunicação.

A organização *Distributed Management Task Force* (DMTF), formalmente conhecida como *Desktop Management Task Force* (DMTF) desenvolveu o *Desktop Management Interface* (DMI) até 1998. Desde aí, que o DMTF é globalmente conhecido pelo seu

[1] As especificações do SNMPv3 fornecem grande capacidade de autenticação mas pouca privacidade, trabalho feito para suportar AES em vez de DES.

trabalho em *Common Information Model* (CIM). A Figura 1 mostra a norma *Lightweight Directory Access Protocol* (LDAP), porque o LDAP foi proposto como uma parte da iniciativa do *Directory Enabled Networks (DEN)*. No inicio 2000 mais uns competidores vieram de dentro do próprio IETF, tais como *Common Open Policy Service for Provisioning* (COPS-PR) [10] e o associado na linguagem de definição de dados *Structure of Policy Provisioning Information* (SPPI) [11].

Em resumo os protocolos de gestão de redes dividem-se em 3 grandes linhas genéricas de orientação: o protocolo normalizado SNMP do IETF, o protocolo normalizado CMIP da OSI e o modelo Standard TMN da ITU-T.

2.3. PROTOCOLO DE GESTÃO DE REDES - SNMP

O protocolo *Simple Network Management Protocol* (SNMP) [1] foi definido no final dos anos 80 para fazer o endereçamento de gestão de redes internet. A arquitectura SNMP tradicional introduz duas entidades: gestores SNMP e agentes SNMP. Os gestores SNMP executam aplicações de gestão SNMP enquanto os agentes SNMP fornecem o acesso às tipologias das variáveis *Management Information Base* (MIB), que são simples escalares e organizadas conceptualmente em tabelas. Os agentes SNMP também podem enviar notificações assíncronas para os gestores SNMP para reportar eventos (*traps*). As tabelas conceptuais e escalares comunicam formalmente via SNMP através do uso de uma linguagem de definição de dados, *Structure of Management Information* (SMI). As definições de dados escritos em SMI são chamadas de módulos de MIB. Ou seja, a norma SMI define como criar uma MIB.

O protocolo SNMP foi desenhado para operar em redes com muitas falhas e para minimizar os requisitos de recursos dos agentes SNMP. Os agentes SNMP são por norma independentes do estado (*stateless*) e o protocolo usado por defeito no transporte das mensagens SNMP é *User Datagram Protocol* (UDP). A primeira versão do SNMP, *Simple Network Management Protocol version 1* (SNMPv1) entrou de forma rápida no processo de normalização (*standardization*) e tornou-se uma norma de internet em 1990 (*Internet standard*) [2]. A primeira versão da linguagem de definição de dados *Structure of Management Information version 1* (SMIv1) foi também publicada em 1990 [3] e uma versão mais formal em 1991 [4], [5]. Tanto, SMIv1 como SNMPv1 foram rapidamente

adoptadas pelos fabricantes de dispositivos de redes e de software de gestão de redes. Actualmente está fortemente disseminado e é suportado por quase todos os dispositivos de rede existentes no mercado.

2.3.1. MODELO DE GESTÃO SNMP

O SNMP é um protocolo de gestão definido ao nível de aplicação, é utilizado para obter informações de vários servidores SNMP na forma de agentes espalhados pela rede com uso da pilha protocolar TCP/IP. Para enviar e receber mensagens através da rede, os dados são obtidos através de requisições (*poll*) de um gestor de rede, para um ou mais agentes através dos serviços do protocolo de transporte *User Datagram Protocol* (UDP). O SNMP permite o acompanhamento simples e fácil do estado da rede, em tempo real, podendo ser utilizado para gerir diferentes tipos de sistemas [112], [113], [115], [116].

O SNMP é o nome do protocolo no qual são trocadas as informações de gestão entre a MIB e a aplicação de gestão, como também é o nome do modelo de gestão.

O modelo de gestão SNMP é dividido em várias entidades:

- O Gestor (Plataforma de Gestão de Redes, Aplicação);
- O Agente (Dispositivo, Elemento de Rede, Software);
- O Protocolo (SNMP);
- A Informação de Gestão (MIB).

Todos os elementos de rede precisam de operar convenientemente para que a rede ofereça os serviços para os quais foi projectada. Estes elementos devem possuir um software especial (o agente) para permitir a sua gestão remota de alarmes. O agente permite a monitorização e o controle de um elemento de rede por uma ou mais plataformas de gestão e responde aos seus pedidos. O gestor comunica directamente com os agentes dos dispositivos através do protocolo de gestão.

2.3.2. TIPOS DE COMUNICAÇÃO ENTRE GESTOR E AGENTE

A comunicação entre o gestor e os agentes dos dispositivos, pode ser de duas formas:

- ***Polling* (pedido, requisição):** É um processo em que o gestor analisa continuamente os seus agentes para obter informação (*poll*). O gestor toma a iniciativa da comunicação e requisita situação do estado actual ao agente (*status).*

- ***Traping* (notificação):** Processo no qual o agente toma a iniciativa de enviar ao gestor a situação do estado actual de determinados eventos anormais que surgem no momento, na forma de uma notificação (*trap*).

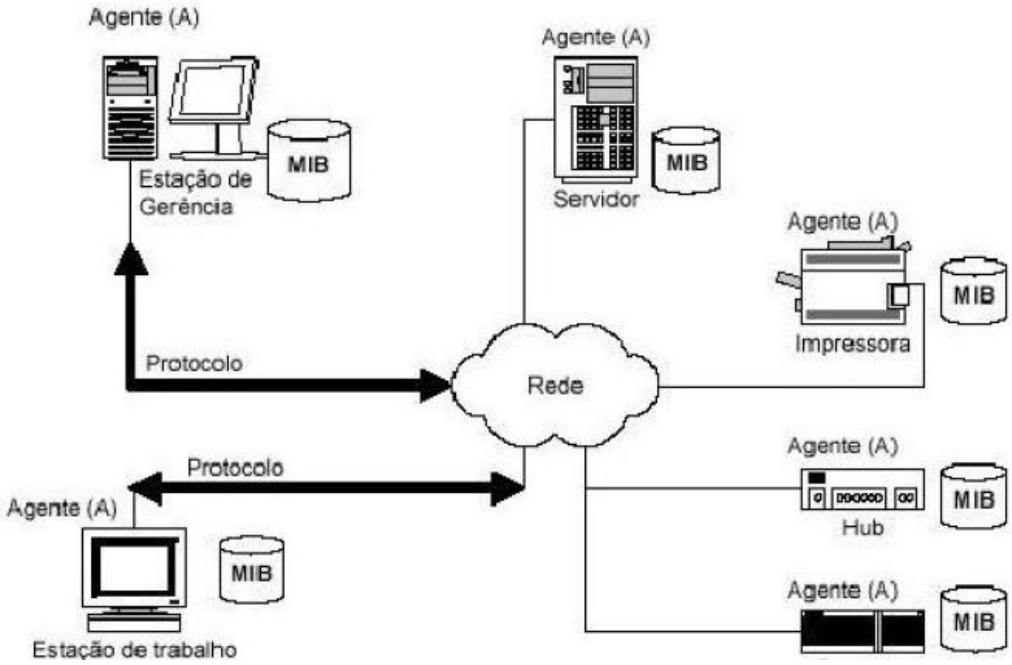


Figura 2 Sistema de Gestão de Rede – Arquitectura Geral.

O protocolo de gestão SNMP define as mensagens usadas entre o gestor e os agentes para trocar informação. Permite operações de monitorização (*Read*) e operações de controlo (*Write*). Esta forma de obtenção de valores de um objecto e suas variações bem como de alteração de um valor de um objecto, com base na utilização de um número muito limitado de operações, torna este protocolo muito simples, estável, flexível e de fácil implementação.

2.3.3. INFORMAÇÃO DE GESTÃO SNMP

O SNMP suporta vários tipos de operações e para cada uma delas existe um *Protocol Data Unit* (PDU) especificado no SNMP [119], [120], das quais destacamos as mais importantes:

- **GetRequest:** A operação *GET* é utilizada para ler o valor da variável; o gestor solicita que o agente obtenha o valor da variável (operação *Read*).

- **GetNextRequest:** A operação de *GET-NEXT* é utilizada para ler o valor da próxima variável; o gestor fornece o nome de uma variável e o cliente obtém o valor e o nome da próxima variável; também é utilizado para obter valores e nomes de variáveis de uma tabela de tamanho desconhecido. O gestor vai procurar interactivamente sequências de informação (operação *Read*).

- **SetRequest:** A operação *SET* é utilizada para alterar o valor da variável; o gestor solicita que o agente faça uma alteração no valor da variável (operação *Write*).

- **Trap:** O agente notifica (sem ter sido solicitado) o valor de uma variável ao gestor. A operação *TRAP* é utilizada para comunicar um determinado evento. O agente comunica ao gestor o acontecimento de um evento, previamente determinado.

  Foram determinados sete tipos básicos de notificação (*trap*): a) *coldStart*: a entidade que a envia foi reinicializada, indicando que a configuração do agente ou a implementação pode ter sido alterada; b) *warmStart*: a entidade que a envia foi reinicializada, porém a configuração do agente e a implementação não foram alteradas; c) *linkDown*: a conexão da comunicação foi interrompida; d) *linkUp*: a conexão da comunicação foi estabelecida; e) *authenticationFailure*: o agente recebeu uma mensagem SNMP do gestor que não foi autenticada; f) *egpNeighborLoss*: um par *Exterior Gateway Protocol* (EGP) perdeu vizinhança; g) *enterpriseSpecific*: indica a ocorrência de uma operação TRAP, não básica.

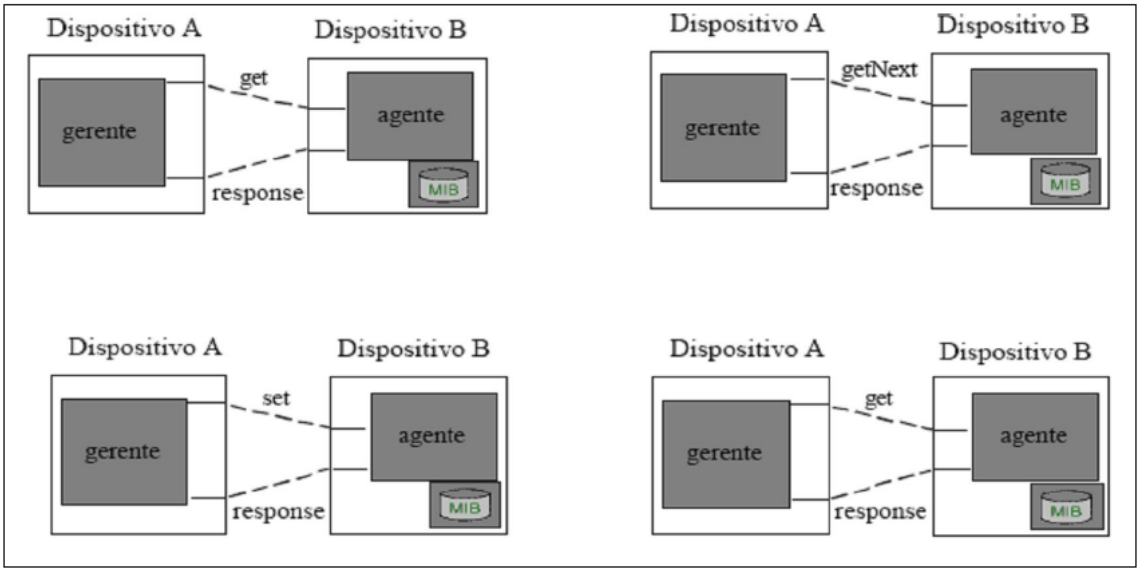


Figura 3 Operações entre gestor e agentes.

A consequência da simplicidade do protocolo SNMP, é a existência de uma redução substancial no tráfego de mensagens de gestão através da rede, que permite a introdução de novas características. Cada máquina gerida é vista como um conjunto de variáveis que representam informações referentes ao seu estado actual, estas informações ficam disponíveis ao gestor através de consulta e podem ser alteradas por ele. Toda a inteligência do processo fica no gestor permitindo que o agente seja uma aplicação muito simples e com o mínimo de interferência no dispositivo em que está a ser executada. As decisões tomadas na ocorrência de problemas e as funções de criar relatórios, ficam sob responsabilidade do gestor.

2.3.4. TIPOS DE AGENTES SNMP

Os agentes SNMP podem ser classificados em dois tipos distintos, que diferem entre si pela forma como são implementadas as funcionalidades do protocolo SNMP e pelo modo como são feitas as interacções com os dispositivos geridos [51]. O primeiro tipo de agente SNMP, é o agente extensível. Este tipo de agente oferece geralmente suporte à *Management Information Base version 2* (MIBv2), e utiliza o SNMP directamente. Um exemplo de um agente extensível é o agente SNMP do ambiente operacional *Microsoft*

Windows. Este agente não contém suporte a nenhuma MIB e para responder às requisições dos objectos de uma determinada MIB, é necessário haver uma biblioteca adicional que implemente o suporte à MIB. Para que o agente extensível comunique com o dispositivo gerido, é necessário a implementação de agentes estendidos. No ambiente operacional das plataformas Unix e Linux pode-se utilizar o agente *Net-SNMP* (anteriormente chamado de UCD-SNMP) [52]. Este agente é usado como base para a implementação de uma grande variedade de agentes estendidos e é um dos mais difundidos neste tipo de ambiente. O agente utiliza a MIB para manipular (obter/alterar) informações do dispositivo com ordens vindas do gestor via SNMP.

2.4. BASE DE INFORMAÇÃO DE GESTÃO - MIB

O protocolo SNMP está mundialmente disseminado pelas redes actuais e é usado em todo o tipo de aplicações que envolve gestão de redes, entre várias tarefas, o SNMP serve para o seguinte tipo de funções: a) monitorizar; b) controlar dispositivos de rede; c) colectar estatísticas; d) gerir configurações; e) medir performance; f) activar os sistemas de segurança; etc. A *Management Information Base* (MIB) é uma forma de troca de informação entre as plataformas de gestão (Gestor) e os agentes nos dispositivos, através do protocolo SNMP. As MIBs são uma das mais importantes e visíveis interfaces de gestão para os dispositivos de rede [113], [115], [116]. A MIB corresponde ao conjunto de informações de gestão disponíveis no agente e define os dados referenciados nas operações do protocolo de gestão, durante a comunicação entre o gestor e o agente. Por outras palavras, o conjunto de todos os objectos SNMP, é colectivamente conhecido como a MIB. É um arquivo codificado de forma que o gestor saiba quais as informações que podem ser solicitadas ao agente e as informações de notificação (*traps*) que poderão ser enviados pelo agente. Todos os objectos acedidos pelo SNMP devem ter nomes únicos, definidos e atribuídos. O gestor e o agente devem acordar os nomes e os significados das operações. A norma do SNMP não define a MIB, mas apenas o formato e o tipo de codificação das mensagens. A especificação das variáveis MIB, tal como o significado das operações em cada variável, são especificados por um padrão próprio.

A árvore MIB "*MIB tree*" é constituída por uma estrutura ramificada que contém as variáveis de gestão de um determinado equipamento, a MIB define para cada variável, um identificador único, denominado *Object Identifier* (OID), formado por um inteiro não negativo. Para localizar uma determinada informação, o identificador da variável (OID),

que será acedido pelo SNMP é representado com o IP do equipamento em questão juntamente com o identificador do objecto na árvore MIB. O OID de um determinado nó da árvore descrita por uma MIB é composto pelo OID do seu pai mais o seu próprio identificador relativo. Entretanto, o uso de números nos OIDs dificulta a compreensão dos nós da MIB e por isso o OID pode ser substituído por um nome (OID *Name*), que pode ser usado em conjunto com o OID numérico. Por exemplo, o OID "1.3.6.1.2.1.1" pode ser representado pelo OID Name "*system*". Por sua vez o "*sysUpTime"* é o OID "1.3.6.1.2.1.1.3", ou "*system.3"*. Os nós da árvore MIB podem ser de diferentes tipos de dados (inteiros, *strings* ou contadores). Também é possível a definição de tabelas, juntamente com a definição do que consta em cada linha da tabela. Por fim, é possível inserir na árvore da MIB informação sobre as *traps*, que podem ser enviadas pelo agente ao gestor, de modo que, o gestor possa interpretar as notificações que ele recebe.

Concluindo, a definição dos objectos da MIB é feita com um esquema de nomes do *Abstract Syntax Notation One* (ASN.1), o qual atribui a cada objecto, um prefixo longo que garante a unicidade do nome (a cada nome é atribuído um número inteiro). O SNMP não especifica conjuntos de variáveis e a definição de objectos é independente do protocolo de comunicação, permitindo desta forma criar novos conjuntos de variáveis MIB, definidos como norma (*standards*), para novos dispositivos ou novos protocolos. Por esta razão, foram criados muitos conjuntos de variáveis MIB que correspondem a protocolos como UDP, IP ou ARP, assim como variáveis MIB para hardware de rede como Ethernet, FDDI ou para dispositivos tais como *routers*, *switches* ou impressoras.

2.5. FERRAMENTAS DE GESTÃO SNMP

O objectivo da gestão de redes é garantir que os utilizadores tenham acesso aos serviços de que necessitam e com a qualidade esperada. Por isso a gestão de redes é vocacionada de forma a garantir: a) sua própria disponibilidade; b) reduzir os custos operacionais; c) aumentar a flexibilidade de operação e integração das redes; d) permitir facilidades de uso; e) garantir características de segurança. Por isso, a actividade de gestão de rede consiste na monitorização de uma rede de comunicações, a fim de diagnosticar problemas e colectar dados estatísticos. As actividades básicas da gestão de redes consistem na detecção e correcção de falhas, num tempo mínimo, e deve estabelecer procedimentos para a previsão de problemas futuros. A complexidade da gestão de rede é directamente proporcional ao tamanho da rede a gerir.

Pelo carácter de livre utilização, será examinado mais a frente a ferramenta *Open Source* de monitorização de redes com protocolo SNMP, chamada NMSIS. Depois serão estudadas as suas caracteristicas e aspectos mais relevantes na actividade de gestão de redes. Existem actualmente várias ferramentas comerciais e de fonte aberta no mercado, mas o custo elevado da activação das licenças comerciais tornam os produtos menos apetecíveis para as pequenas e médias empresas e até mesmo para algumas das grandes empresas que têm vindo, pouco a pouco apostar cada vez mais em ferramentas com código de fonte aberta. Recorrendo ao modelo FCAPS da ISO [49], [114], [121], podemos dividir as necessidades de gestão em vários aspectos distintos:

a) Na gestão de falhas, é preciso saber distinguir falha e erro. Uma falha é uma condição anormal que requer uma acção para a sua correcção, por exemplo, se uma linha de comunicação for cortada fisicamente, o sinal não passa na linha. Enquanto o erro é um evento simples, como uma torção no cabo ou outro tipo de anomalia que pode causar distorções, que induzem a uma taxa elevada de erros.

Para que a rede em caso de falha fique operacional o mais rápido possível, torna-se imperativo seguir os seguintes passos no processo de gestão de falhas:

- Detectar a falha;
- Isolar a falha;
- Restaurar o serviço;
- Identificar as causas do problema;
- Resolver o problema.

b) A gestão de desempenho de uma rede, é a monitorização das actividades de rede e o controlo dos recursos através de ajustes e alterações de configuração. Algumas das questões relativas à gestão de desempenho são:

- O nível de capacidade de utilização;
- A quantidade de tráfego (se tráfego é excessivo);
- Se o *throughput* foi reduzido para níveis aceitáveis;

- Se o tempo de resposta está aumentar.

c) A gestão de configuração está relacionada com as tarefas de manutenção, adição e actualização dos componentes e do estado dos componentes durante a operação de rede. A gestão de configuração engloba também a topologia de rede, o mapeamento da rede e inclui ainda o planeamento e o projecto de rede. Ao nível dos serviços, estes devem ter disponíveis os seguintes parâmetros: tempo de resposta; taxa de rejeição e disponibilidade.

d) A gestão de contabilização permite que o administrador determine se um utilizador ou grupo de utilizadores, estão abusar dos seus privilégios de acesso, se os utilizadores estão a usar a rede de forma ineficiente e a consumir muita banda na rede.

e) A gestão de segurança permite que sejam tomadas políticas de segurança para garantir a monitorização e o controle de acessos à rede, ou parte da rede, às informações de colecta, ao armazenamento e análise de registos de *logs* de segurança.

O uso do SNMP é bastante interessante na medida em que permite que se utilize inúmeras ferramentas de gestão. O âmbito de estudo deste projecto visa a integração do sistema NMSIS que utiliza o protocolo SNMP com um sistema de correlação de eventos. Pretende-se dotar o NMSIS com um motor de correlação que permita dar uma resposta mais adequada aos eventos que diariamente povoam o sistema de gestão de redes. Tal como o NMSIS, existem outras ferramentas com protocolo SNMP, já que a facilidade do uso do SNMP permite que se crie cada vez mais ferramentas e opções para a gestão de equipamentos. O potencial das ferramentas *Open Source* é imenso, desde a facilidade no desenvolvimento, documentação disponível, suporte e claro os custos inexistentes com software do tipo *freeware*.

Como o sistema NMSIS não têm nenhum módulo que faça correlação de eventos, o âmbito de estudo deste trabalho de Tese de Mestrado centra-se nessa vertente. O próximo capítulo, fará análise do sistema NMSIS, para futura inclusão de um sistema integrado de gestão e correlação de eventos.

2.6. NMSIS

O *Network Management System with Imaging Support* (NMSIS) foi criado para dar resposta às necessidades de gestão de uma LAN. O NMSIS é uma aplicação de gestão e monitorização de redes informáticas, que resulta da integração das ferramentas *Network Administration Visualized* (NAV), *Free Open Source Ghost* (FOG) e *Bandwidth Traffic Control* (BWTC), representadona figura 4. Entre outras valências no ramo de gestão de redes, o NMSIS contém na sua estrutura outras funções desenpenhadas por ferramentas de instalação de software em computadores remotos através da rede, denominada de gestão de imagens de software. Esta função é realizada através da ferramenta FOG. O NMSIS posuí uma outra ferramenta de controlo de tráfego, chamado BWTC. O BWTC molda o fluxo de tráfego do servidor de imagens para vários clientes, de acordo com a taxa de utilização da rede.

Entre as diversas valências de gestão do NMSIS destacam-se: uma interface Web de gestão com delegação de tarefas para diferentes gestores; a localização de equipamento na rede; o inventário da rede; gráficos de performance para os equipamentos de rede; ferramentas de apoio à manutenção; a monitorização de equipamento (por exemplo, switches, routers, servidores, etc.); a monitorização de serviços (por exemplo, o serviço de páginas de internet, o serviço de mail, etc.); envio de alertas por correio electrónico quando ocorrem eventos; gestão de imagens de software; entre muitas outras [111]. Este sistema forma um universo constituído por várias aplicações.

Vamos começar por focar o estudo na fracção integrante do NMSIS que é responsável pelas funções de pooling aos elementos de rede e de que forma essa informação é armazenada pelo sistema. A base de dados do NMSIS (NMSISdb) tem como principal função, o armazenamento de todos os parâmetros de rede. Foi elaborado um estudo ao sistema integrante do NMSIS, responsável pela monitorização e gestão de rede, descrito na figura 4 [168], com o nome de NAV.

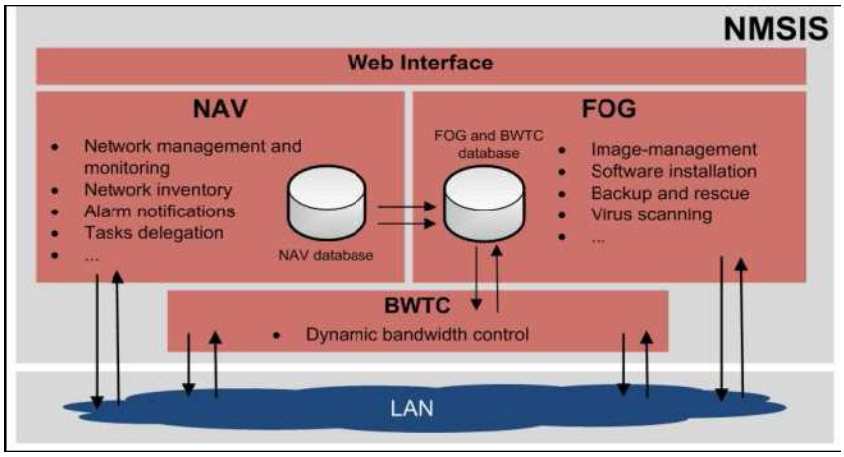


Figura 4 Arquitectura do NMSIS.

Características e Funcionalidades Especificas:

- Suporta SNMP;
- Inventário dos elementos da rede (versão de software, local, tipo, etc);
- Monitorização de serviços de rede (serviços WEB, MAIL, Bases Dados, Rede);
- Monitorização dos recursos do computador (Carga de processamento, utilização de disco, etc.);
- Acesso remoto via SSL ou SSH;
- Processo de escalonamento dos problemas na notificação;
- Administração de segurança e acessos à ferramenta de gestão;
- Monitorizar e medição tráfego nas ligações de rede com geração de alertas;
- Notificações por E-MAIL, SMS e outras definidas pelo utilizador;
- Interface com utilizador via WEB;
- Base de dados que guarda informação de rede;

- Informação da topologia de rede, interconexão entre elementos é automática;
- Tem monitorização de estado que aponta para possível causa do problema;
- Estatísticas de tráfego, contadores de pacotes e contadores de erros;
- Histórico do movimento das máquinas na rede;
- Ferramentas de manutenção e logística;
- Licenciamento segundo as regras GNU - *General Public License*;
- Boa documentação de apoio no desenvolvimento;
- Faz armazenamento de dados;
- Tem inúmeras ferramentas de font end;
- Tem processos de back end.

Observações Gerais:

Apresenta a versão de software de todos os equipamentos, o tipo de equipamento, a sua localização e uma interface gráfica para a utilização de portos de *switches* e *routers* (por exemplo, se as portas estão em *trunk*, livres, ocupadas, bloqueadas, etc.). Adicionalmente, disponibiliza uma ferramenta para bloquear as portas de *switches* via SNMP. A interligação dos elementos de rede (*Routers*, *Switches*, Servidores ou computadores), é de certa forma auto-detectável. Dispõe de um serviço que faz a monitorização do estado da rede e que detecta quando o tráfego nas intersecções da rede ultrapassa o valor definido pelo gestor (por exemplo, X% de taxa de utilização). O NMSIS fornece estatísticas de utilização para vários elementos de rede, tais como servidores, routers e *switches*, desde que estes suportem o protocolo SNMP. Disponibiliza um mapa de tráfego que mostra a topologia da rede ao nível da camada OSI de Rede, e ao nível da camada de ligação lógica. A monitorização de carga de CPU, utilização de memória, tráfego nas intersecções de rede, entre outros, pode ser feita de forma permanente. E caso se atinja determinados valores de *threshold* são lançados alarmes (por mail ou SMS). Com o NMSIS é igualmente possível monitorizar o estado de operacionalidade dos servidores, computadores, activos de rede e respectivos serviços (HTTP, POP, SMTP, SMB, etc.), impressoras de rede, entre outros.

2.6.1. MONITORIZAÇÃO DE EVENTOS E NOTIFICAÇÕES DO NMSIS

Para o desenvolvimento deste trabalho, interessa-nos focar nos eventos de falhas de rede, mais do que em qualquer das outras características que o sistema possui. Para podermos saber quais os eventos de falhas de rede que são gerados pelas ferramentas do NMSIS e como poderemos colectá-los no sistema de correlação. É preciso entender como todas as tabelas da base de dados NMSIS se conjugam, para podermos trabalhar esses eventos.

2.6.2. BASE DE DADOS

A base de dados principal [111] é a "*manage*" e contém a informação de topologia da rede, tabelas para o sistema de eventos e alertas, para o sistema de mensagem, etc. A base de dados de perfis, tem informação dos utilizadores e seus perfis. A base de dados "*Arnold*" tem informação das portas dos *switches* que estão bloqueadas pela ferramenta de *front end* do "*Arnold*". A base de dados "*logger*", é uma pequena base de dados para o "*Cisco syslog analyzer*".

2.6.3. TABELAS PRINCIPAIS DO NMSIS, COMO SE CONJUGAM

Importa agora conhecer a base de dados "*manage*", conhecida por *NMSISd*, que faz a gestão de eventos de todo o sistema. Esta base de dados é bastante extensa. A sua tabela principal é a "netbox", para quem concebeu a *NMSISdb*, esta tabela é o coração do sistema [144].

Tabela 1 Tabela *netbox*

netboxid	primary key
ip	IP address of the netbox
roomid	room the box is placed in
deviceid	the device this is (foreign key to device)
typeid	sysobjectid of the box
sysname	name of the box, based on fully qualified dns name with fall back to IP address.
catid	category of the box (GW, SW, SRV, etc)
subcat	DEPRECATED? - check before delete!!!
orgid	organization that manages the box
ro	snmp read community
rw	snmp write community
prefixed	prefix the netbox is on
up	whether the box is up and running (as seen from pping)
snmp_version	version of the snmp agent (1 or 2)
snmp_agent	DEPRECATED? - check before delete!!! Was used by a server collection module once.
upsince	the timestamp when the box was last booted. Data is taken from mibII system.uptime
uptodate	whether gDD has done OID classification
discovered	timestamp when the box was first discovered by NMSIS

I. Tabela *netbox* - *C*onjugada com outras tabelas

A tabela *netbox* é a mais central de todo o sistema, contém informação de todos os dispositivos IP que o NMSIS gere. Quando se adiciona informação na tabela, existe relações entre a informação adicionada, porque existe uma serie de relações com outras tabelas [144].

Tabelas que mantêm uma relação com a *netbox*:

- A tabela *netboxinfo* que é o local onde se grava informação adicional da *netbox*;
- A tabela *device* contém informação de todos os dispositivos físicos da rede, ao contrário da *netbox* esta tabela tem como foco principal a "*box*" física dos dispositivos, como por exemplo o *serial number*;
- A tabela *module*, define os módulos que são partes da tabela *netbox* e da coluna *category*, são os portos e dispositivos com *serial number*;
- A tabela *mem* que descreve a memória e *nvram* da *netbox*; a tabela *room* que define locais técnicos, tais como sala de servidores ou sala rede, etc;
- A tabela *location* que define grupos de *room*, tais como campus, DEG, DEE, etc;
- A tabela *org* que define a organização;
- A tabela *cat* que define categorias da *netbox* tais como *GW, SW, SRV, OTHER;*
- A tabela *subcat* que define subcategorias dentro da categoria;
- A tabela *netboxcategory* que pode estar em várias subcategorias, essa relação é definida aqui;
- A tabela *type* que define o tipo de *netbox*; e a tabela *vendor* que define os fabricantes.

Podemos constatar isso mesmo no esquema da figura 5.

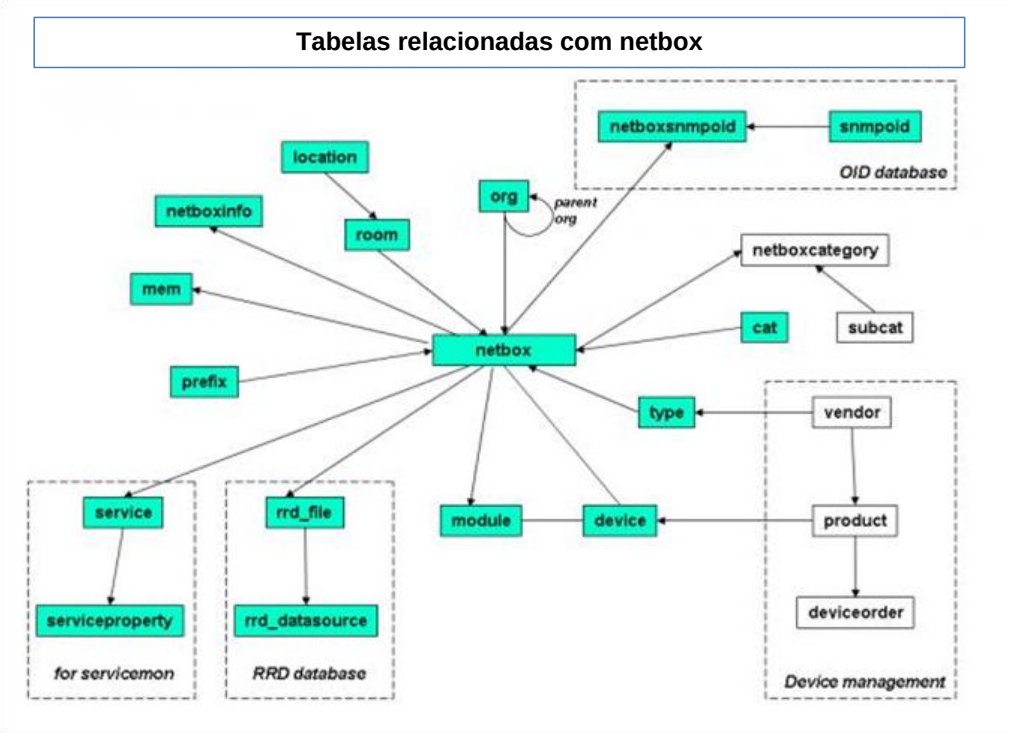


Figura 5 Relações entre várias tabelas e a *netbox* da NMSISdb.

II. Tabelas de topologia dos *routers* conjugadas

Existe uma serie de relações entre várias tabelas com informação de topologia de rede. Os routers através de uma serie de protocolos, são capazes de ter uma base de dados topológica das suas interligações [144].

As tabelas que se conjugam para formar uma tipologia de rede de camada 3**:**

- A tabela *gwport* que define as portas do router que estão ligadas a determinado módulo, apenas as portas que tem um endereço IP configurado e estão em "*not shutdown"* são consideradas nesta tabela; a tabela *gwportprefix* define os endereços da porta do router, uma ou mais;
- A tabela *prefix* guarda os prefixos dos endereços IP;
- A tabela *vlan* define o domínio de broadcast que tem um valor de vlan;
- A tabela *nettype* define o tipo de rede, tais como *lan*, *core*, *link*, *elink*, *loopback*, *closed*, *static*, *reserved*, depois de definidos no *NMSIS* o valor nunca deve ser alterado;

- A tabela *usage* define um grupo de utilizadores;
- A tabela *arp* contém o IP e o MAC da tabela de arp dos routers e respectivo *timestamp*.

Como mostra a figura 6, podemos verificar a interacção entre essas tabelas.

III. **Tabelas de topologia dos *switches* conjugadas**

Não só os routers permitem ter esse mapa topológico, também através dos switches é posivel ter informação topológica da rede [144].

As tabelas que se conjugam para formar uma tipologia de rede de camada 2**:**

- A tabela *swport* que define as portas do switch que estão ligadas a determinado modulo;
- A tabela *swportvlan* que define os valores das vlans em todos os portos dos *switches*;
- A tabela *swportallowedvlan* que guarda uma *string* que representa uma serie de vlans que são permitidas atravessar determinada porta do switch;
- A tabela *swportblocked* que define quais as portas que estão bloqueadas pelo protocolo de camada 2 *Spanning Tree Protocol* (STP), que previne a existência de *loops* na rede;
- A tabela *swp_netbox* que é usada no processo de construção da topologia fisica, define os candidatos ao próximo salto físico;
- A tabela *netbox_vtpvlan* que contém a informação do protocolo *Vlan Trunking Protocol* (VTP) da base de dados do *switch*;
- A tabela *cam* que define a porta do *switch* e respectivo *MAC Adress* e o *timestamp*.

Podemos verificar a interacção entre essas tabelas, na figura 6.

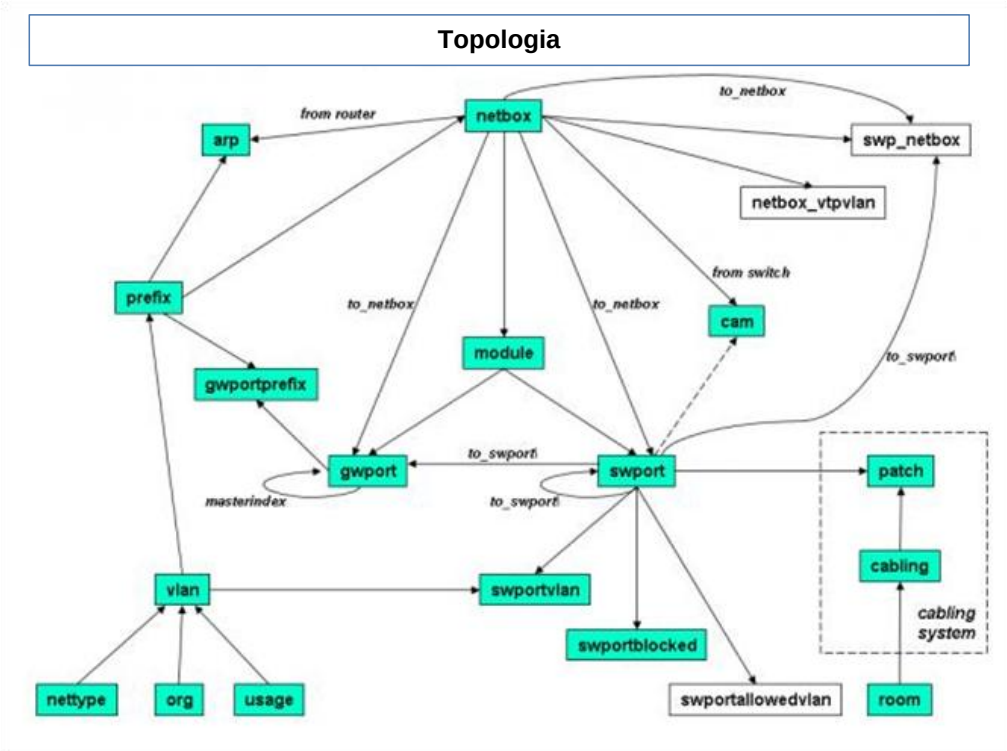


Figura 6 Tabelas de topologia conjugadas dos Router e Switches

IV. Tabelas do sistema de eventos conjugadas

Existe também uma serie de relações entre várias tabelas que fazem parte do sistema de eventos [144].

As tabelas conjugadas para formar um sistema de fluxo de eventos**:**

- A tabela *eventq* (*event queue*) que tem dados adicionais na tabela *eventqvar*, tem informação de eventos colocados por diferentes subsistemas especificados na fonte, normalmente o "*event engine"* é o destino de processamento dos dados do "*event queue*";
- A tabela *eventtype* define tipos de eventos;
- A tabela *subsystem* define vários subsistemas que colocam (*post*) ou recebem os eventos;

- A tabela *eventqvar* define alguns valores adicionais (*key, value)* que seguem junto com os eventos;

- A tabela *alertq* tem informação adicional na tabela *alertqvar* e *alertmsg*, o "*alert engine*" coloca eventos no "*alert queue*" e paralelamente na tabela *alerthist*. O "*alert engine*" processa os dados do "*alert queue*" e envia alertas para utilizadores com base nos seus perfis de alerta. Quando todos os utilizadores registados tiverem recebido o alerta, o "*alert engine*" irá remover os alertas da tabela *alertq* (mas não no "*alert history"*)

- A tabela *alerttype* define os tipos de alertas, um tipo de evento pode ter vários tipos de lertas;

- A tabela *alertqmsg*. O "*event engine*" tem por base o *alertmsg.conf*, que é um pré-formato de mensagens de alarme, ou seja, uma mensagem para cada canal de alerta configurado (mail ou sms), uma mensagem para cada linguagem configurada. Os dados são guardados na tabela *alertqmsg*;

- A tabela *alerthist* que é similar a tabela "*alert queue*" com uma importante distinção, a "*alert history*" guarda o estado dos eventos como uma linha, com o timestamp (*start time* e *end time*) do evento;

- A tabela *alerthistvar* define alguns valores adicionais (*key, value)* que seguem junto com as gravações da *alerthist*;

- A tabela *alerthistmsg* serve para também ter um histórico das mensagens formatadas.

Podemos verificar todas estas conjugações na figura 7.

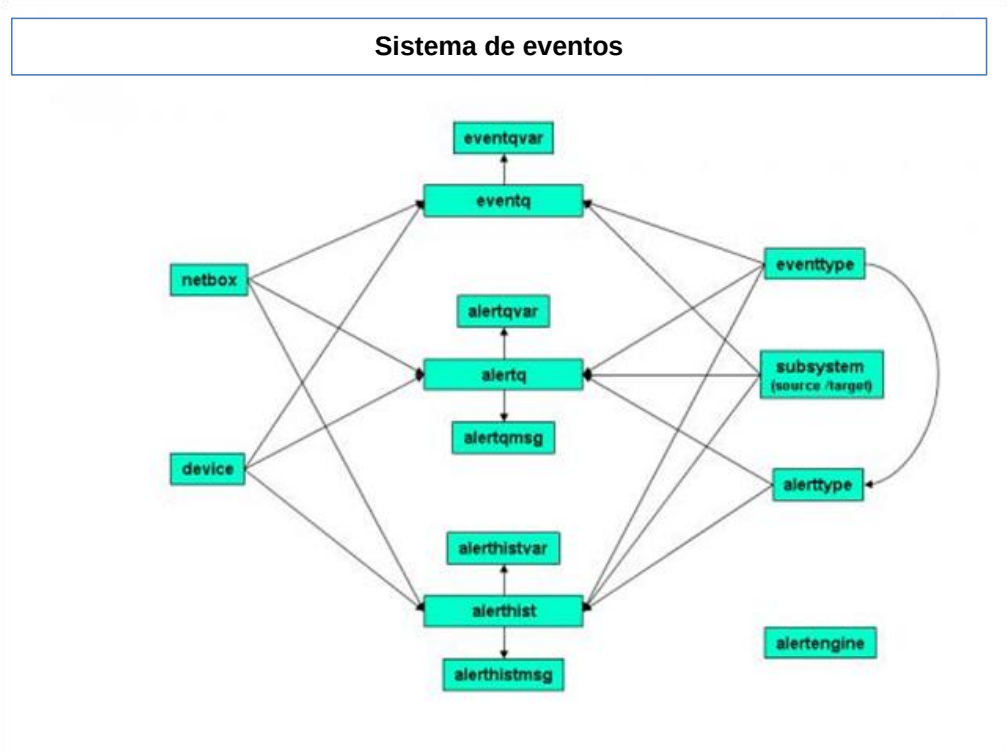


Figura 7 Tabelas de eventos e alertas conjugadas entre si.

O NMSIS dispõe de várias ferramentas que interagem com as tabelas da base de dados "*manage*". Essas ferramentas de *front end* descritas normalmente em linguagem de programação Python, são responsáveis por introduzir e modificar os valores de determinados campos específicos das tabelas de forma a a fazer a monitorização da rede.

2.6.4. AS FERRAMENTAS DE ESTADO DO NMSIS (THE STATUS TOOL)

O NMSIS dispõe da ferramenta Status que fornece uma perspectiva global da operação e da actividade dos alarmes, incluindo os componentes de serviço da rede que estão em produção. O NMSIS dispõe de quatro processos a correr em *background* que monitorizam o estado operacional da rede. Estes lançam eventos com base na informação que recolhem e que fica disponível na ferramenta Status. Os processos são: a) o ***status monitor*** (*pping* [144]), que verifica se os equipamentos estão alcançáveis na rede; b) o ***service monitor*** (*servicemon* [144]) que monitoriza os serviços; c) o ***module monitor*** (*modulemon* [144]) que monitoriza a operação dos módulos dentro dos *switches*; d) e o ***threshold monitor*** (*thresholdmon* [144])

que monitoriza os dados RRD e envia alertas se os valores limites definidos forem excedidos.

Existe ainda o processo ***snmptrapd*** que apenas recolhe *traps* enviadas pelos equipamentos de rede. Todas estas ferramentas encaminham os dados para o *event queue*. A Figura 8, fornece uma perspectiva geral do funcionamento da ferramenta Status.

Destas quatro ferramentas, as duas primeiras são as que nos interessam estudar porque são as que gerem os alarmes dos eventos activos de rede bem como dos seus serviços. A análise do processo *snmptrapd* que recolhe *traps* enviadas pelos equipamentos de rede é também importante para o estudo deste trabalho.

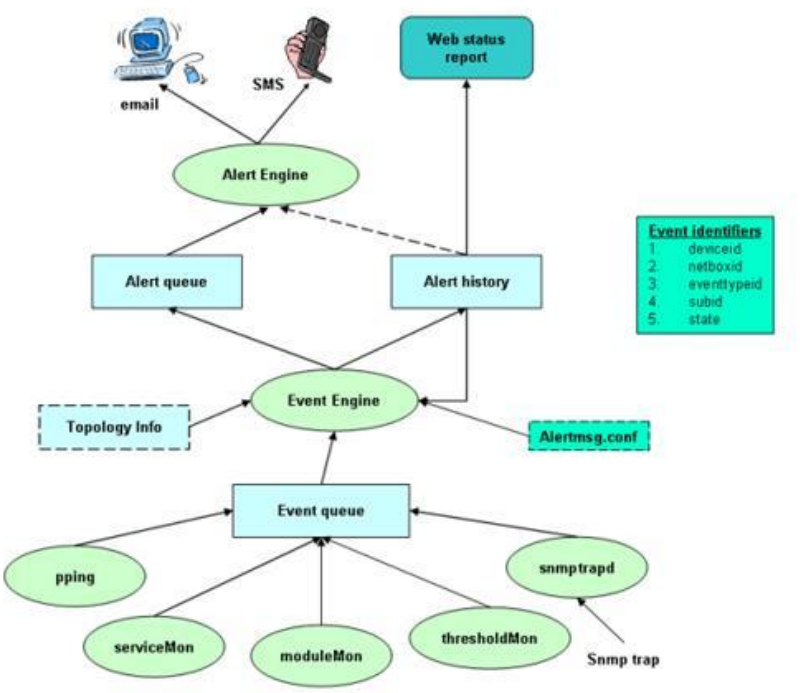


Figura 8 Ferramenta Status do NMSIS

I. A Ferramenta de estado "status monitor" (pping - parallel pinger):

A ferramenta *status monitor* é feita através do "*pping"*, que é um *daemon* com o seu próprio mecanismo de agendamento (configurável). A frequência de cada varrimento aos elementos de rede via *ping* é por defeito, vinte segundos. O tempo de resposta máximo

permitido por elemento de rede, é por defeito cinco segundos. Um elemento é considerado inatingível (em baixo) e colocado na tabela *event queue* após quatro "*no responses*" consecutivas (também configuráveis). São precisos cerca de oitenta a noventa segundos para o *pping* detectar que o elemento está efectivamente em baixo.

Este *daemon* pinga todas as "*boxes*" que são elementos activos da rede da tabela *netbox* e como trabalha em paralelo com o sistema, é possível pingar grandes quantidades de "*boxes*". Tem dependência de ter todos os elementos dentro da tabela *netbox*, para fazer a sua tarefa. Esta ferramenta com linguagem de programação *Python*, insere os eventos na tabela *event queue* e insere o valor na tabela e coluna (*netbox.up*).

A tabela *event engine* tem um período de cerca de um minuto (configurável) até o evento "*box down warning*" ser fixado na tabela *event queue.* E serão precisos mais três minutos para a "*box*" ser declarada "em baixo" (também configurável). Em suma, é preciso esperar cinco a seis minutos até o elemento ser declarado "em baixo". O ficheiro de configuração é "*pping.cnf*" e permite ajustar determinados parâmetros, como mostra a tabela 2.

Tabela 2 Parâmetros ajustáveis no ficheiro pping.conf.

parameter	description	default
user	the user that runs the service	NMSIScron
packet size	size of the icmp packet	64 byte
check interval	how often you want to run a ping sweep	20 seconds
timeout	seconds to wait for reply after last ping request is sent	5 seconds
nrping	number of requests without answer before marking the device as uNMSISailable	4
delay	ms between each ping request	2 ms

II. A Ferramenta de estado "*service monitor*" (*servicemon - The service monitor)*:

A ferramenta *service monitor* (*servicemon),* tal como a anterior "*pping"* é um *daemon* que faz o *polling* da rede. Tem um ficheiro de configuração "*servicemon.conf*", em linguagem de programação Python. Esta ferramenta monitoriza as netboxes, usa os *handlers* implementados na rede (tipo de serviço) para lidar com o número crescente de serviços na rede. Permite vários tipos de serviço, actualmente suporta *ssh, http, imap, pop3, smtp, samba, rpc, dns and dc*. Tem dependência de ter as tabelas *service* e *serviceproperty*

prenchidas com dados. Isto é prenchido pelo "*Edit Database*" quando o admnistrador NMSIS regista os serviços que pretende monitorizar. Faz *update* da tabela *eventq* com eventos *servicemon.*

Esta aplicação executa uma rotina, que verifica cada serviço a cada sessenta segundos. Tem vários *timeouts* para os diferentes tipos de serviço e variam entre cinco e dez segundos. Se o serviço não responde três vezes numa linha, o "*servicemon"* declara o serviço "em baixo".

III. **Ferramenta *snmptrapd (The SNMP trap daemon)*:**

A ferramenta *snmptrapd* tal como as anteriores, é também um *daemon,* só que não faz *polling*. Fica à escuta no porto 162 do protocolo UDP, à espera de receber notificações (*traps*). Quando *snmptrapd* recebe uma *trap*, põe toda a informação num formato chamado de notificação objecto (*trap-object)* e envia o objecto para todos os *traphandler* que estão configurados na opcção "traphandlers" do "*snmptrapd.conf"*. Fica depois ao critério do *traphandle*r decidir se quer processar a *trap* ou quer discartá-la. A linguagem de programação é o Python e o ficheiro de configuração é o "*snmptrapd.conf"*. O update das tabelas depende dos *traphandler* e os eventos são colocados normalment no *eventq.*

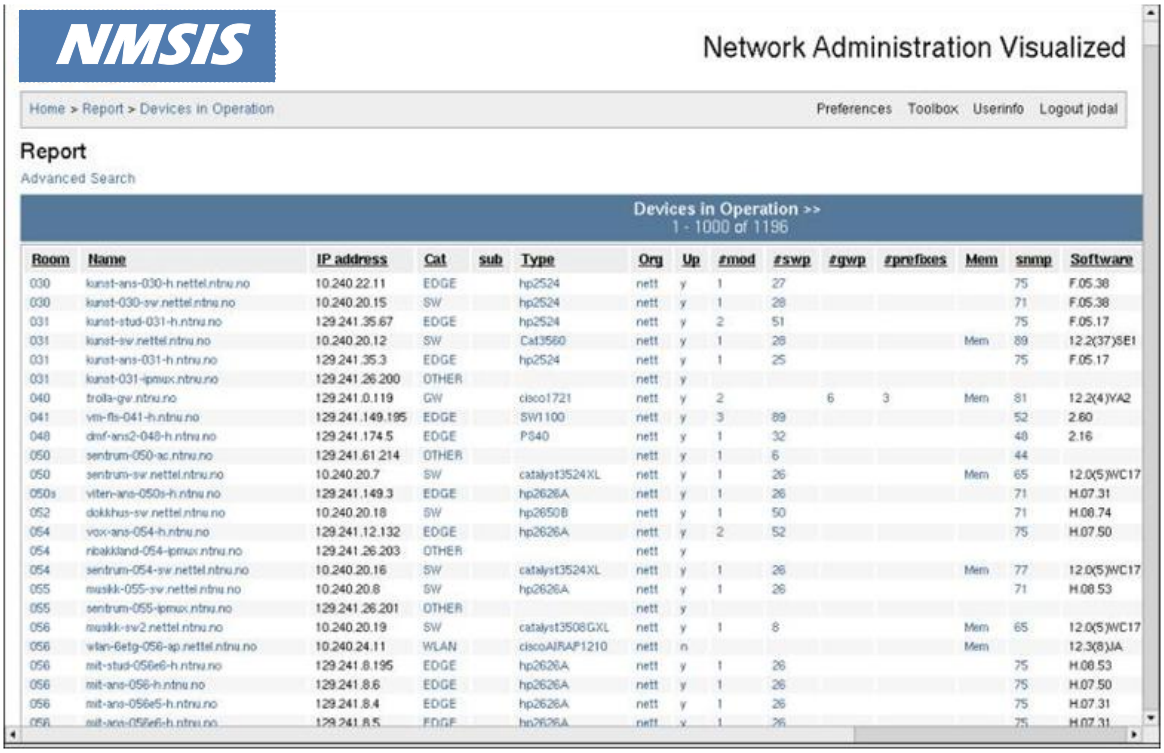
2.6.5. CONSIDERAÇÕES FINAIS SOBRE A FERRAMENTA DE GESTÃO NMSIS

Foi feita uma análise mais exaustiva às tabelas da base de dados do NMSIS, como se conjugam e relacionam. Foram estudados os principios de funcionamento das ferramentas de estado que fazem o *front-end* na monitorização de eventos. Uma das primeiras questões que surgiram na abordagem ao sistema NMSIS para poder ser integrado com um motor de correlação era saber se o sistema de correlação iria actuar como um *front-end* desta solução ou um seria um sistema de recolha de eventos provenientes do NMSIS. Ou seja, se este sistema comunicaria directamente com os agentes dos equipamentos de rede.

A conclusão foi obvia, seria inviável integrar o módulo de correlação como um *front- end,* visto que repetiria as mesmas funções de gestão SNMP, desempenhadas pelo NMSIS. O NMSIS em si já tem todas as funcionalidades para funcionar como *front-end* e como *back-end,* assim do ponto de vista funcional será muito mais coerente gerir todos os recursos da rede e respectivos eventos de uma forma mais global com o NMSIS a gerar todos os eventos de rede para serem analisados à entrada do novo sistema. A uníca forma seria

receber eventos através de notificações (*traps)* directamente dos elementos de rede, configurados para o efeito.

O NMSIS centra toda a sua arquitectura de software em bases de dados relacionáveais para armazenamento de informação. O estado de todos os activos de rede, é feito através da informação guardadas em várias tabelas, essa informação é depois disponibilizada como mostra a figura 9, onde poderemos visualizar vários activos pelo seu endereço de rede e o respectivo estado actual na coluna "*up*", além de outras informações.



Room	Name	IP address	Cat	sub	Type	Org	Up	#mod	#swp	#gwp	#prefixes	Mem	snmp	Software
030	kunst-ans-030-h.nettel.ntnu.no	10.240.22.11	EDGE		hp2524	nett	y	1	27				75	F.05.38
030	kunst-030-sw.nettel.ntnu.no	10.240.20.15	SW		hp2524	nett	y	1	28				71	F.05.38
031	kunst-stud-031-h.ntnu.no	129.241.35.67	EDGE		hp2524	nett	y	2	51				75	F.05.17
031	kunst-sw.nettel.ntnu.no	10.240.20.12	SW		Cat3560	nett	y	1	28			Mem	89	12.2(37)SE1
031	kunst-ans-031-h.ntnu.no	129.241.35.3	EDGE		hp2524	nett	y	1	25				75	F.05.17
031	kunst-031-ipmux.ntnu.no	129.241.26.200	OTHER			nett	y							
040	trolla-gw.ntnu.no	129.241.0.119	GW		cisco1721	nett	y	2		6	3	Mem	81	12.2(4)YA2
041	vm-fts-041-h.ntnu.no	129.241.149.195	EDGE		SW1100	nett	y	3	89				52	2.60
048	dmf-ans2-048-h.ntnu.no	129.241.174.5	EDGE		PS40	nett	y	1	32				48	2.16
050	sentrum-050-ac.ntnu.no	129.241.61.214	OTHER			nett	y	1	6				44	
050	sentrum-sw.nettel.ntnu.no	10.240.20.7	SW		catalyst3524XL	nett	y	1	26			Mem	65	12.0(5)WC17
050s	viten-ans-050s-h.ntnu.no	129.241.149.3	EDGE		hp2626A	nett	y	1	26				71	H.07.31
052	dokkhus-sw.nettel.ntnu.no	10.240.20.18	SW		hp2650B	nett	y	1	50				71	H.08.74
054	vox-ans-054-h.ntnu.no	129.241.12.132	EDGE		hp2626A	nett	y	2	52				75	H.07.50
054	nbakkland-054-ipmux.ntnu.no	129.241.26.203	OTHER			nett	y							
054	sentrum-054-sw.nettel.ntnu.no	10.240.20.16	SW		catalyst3524XL	nett	y	1	26			Mem	77	12.0(5)WC17
055	musikk-055-sw.nettel.ntnu.no	10.240.20.8	SW		hp2626A	nett	y	1	26				71	H.08.53
055	sentrum-055-ipmux.ntnu.no	129.241.26.201	OTHER			nett	y							
056	musikk-sw2.nettel.ntnu.no	10.240.20.19	SW		catalyst3508GXL	nett	y	1	8			Mem	65	12.0(5)WC17
056	wlan-6etg-056-ap.nettel.ntnu.no	10.240.24.11	WLAN		ciscoAIRAP1210	nett	n					Mem		12.3(8)JA
056	mit-stud-056e6-h.ntnu.no	129.241.8.195	EDGE		hp2626A	nett	y	1	26				75	H.08.53
056	mit-ans-056-h.ntnu.no	129.241.8.6	EDGE		hp2626A	nett	y	1	26				75	H.07.50
056	mit-ans-056e5-h.ntnu.no	129.241.8.4	EDGE		hp2626A	nett	y	1	26				75	H.07.31
056	mit-ans-056e6-h.ntnu.no	129.241.8.5	EDGE		hp2626A	nett	y	1	26				75	H.07.31

Figura 9 Tollbox report

Toda essa informação poderia ser disponibilizada na forma de um ficheiro de *log* com todos os eventos produzidos a cada instante. O NMSIS não tem esse tipo de aproximação, que pudesse reportar todos os eventos de rede relacionados com a gestão de todos os activos de rede. Tem sim, um sistema de eventos e alertas internos que reportam falhas intrinsecas do próprio sistema. Tal como um sistema Unix ou qualquer dispositivo de rede como um (*Router* ou *Switch*), essa opção de varrimento de *logfiles* seria interessante no contexto deste trabalho, mas tal não foi possivel.

O objectivo deste trabalho não se prende com análise dos eventos internos do NMSIS, mas sim com os eventos externos, ou seja, os elementos de rede por ele geridos. Foram ponderadas várias formas de como obter a informação dos eventos do NMSIS. Uma das formas possiveis foi criar um *script* (*daemon*) que pudesse ler as tabelas do NMSIS e pudesse ele próprio fazer essa função de criar uma espécie de ficheiro de *log* contendo toda a informação de estado dos objectos (elementos de rede) a cada momento. Esse trablho será exemplificado mais a frente.

Posto isto, é agora possivel trabalhar toda esta informação para poder colectar os eventos que interessam usar à entrada do novo sistema SCEN. O novo motor de correlação terá como função analisar todos estes eventos gerados por esse *daemon*.

2.7. RESUMO DO CAPÍTULO

Após um estudo introdutório aos protocolos normalizados a nível internacional, tais como o SNMP da comunidade IETF, o CMIP da comunidade ISO, o TMN da comunidade ITU e o ILMI da comunidade ATM *Forum. F*oi estudado o contexto histórico de como apareceram. Com o propósito de interligar várias redes heterogéneas a nível nacional e internacional, e conceber um modelo que descrevesse um conjunto de aspectos condições gerais para desenvolvimento de protocolos específicos de rede. Um dos muitos protocolos na camada de aplicação e de grande interesse para o âmbito de estudo desta Tese é o SNMP.

O protocolo SNMP foi definido no final dos anos 80 para fazer o endereçamento de gestão de redes internet. A arquitectura SNMP introduz algumas entidades: os gestores SNMP, os agentes SNMP, o protocolo SNMP e a informação de gestão (MIB). O gestor comunica directamente com os agentes dos elementos de rede através do protocolo de gestão de duas formas distintas de comunicação: via polling e via traping. Existem várias operações que são feitas entre gestor e o agente, das quais apenas citamos as mais importantes e que fazia parte do SNMPv1: *GetRequest, GetNextRequest, SetRequest, Trap*. Os agentes SNMP podem ser qualificados em dois padrões distintos, que diferem entre si pela forma como são implementadas as funcionalidades do protocolo SNMP e pelo modo como são feitas as interacções com os dispositivos geridos: agente extensível e agente estendido. A troca de informação entre as plataformas de gestão (gestor) e os agentes é feita por intermédio da

MIB que corresponde ao conjunto de informações de gestão, disponíveis no agente e define os dados nas operações do protocolo de gestão.

Como o âmbito de estudo deste trabalho de Tese é a introdução de um novo módulo com um Sistema Correlação de Eventos e Notificações (SCEN) num sistema de gestão de alarmes com protocolo SNMP. No caso específico deste trabalho, o enquadramento da ferramenta de gestão SNMP será na ferramenta que já está em produção no departamento DEE e DEG, chamada NMSIS.

O SCEN vai servir para instrumentar e dotar este sistema de gestão NMSIS, com um bloco de correlação de eventos que não existia. Por isso foi feita a análise pormenorizada ao sistema NMSIS, com o objectivo de criar um sistema integrado de gestão e correlação de eventos. Foram estudadas as suas características e funcionalidades mais importantes. Foi estudado a constituição da base de dados NMSISdb, as tabelas mais importantes, as suas relações e o modo com se conjugam. Por fim, foram estudadas as várias ferramentas de estado do NMSIS (*Status Monitor, Service Monitor* e *snmptrapd*), as suas caracteristicas e principais funções.

3. INTRODUÇÃO A FERRAMENTAS COM CORRELAÇÃO DE ALARMES

3.1. INTRODUÇÃO À CORRELAÇÃO DE EVENTOS

Após a conclusão do capítulo anterior, com o estudo do modelo de organização das bases de dados e respectivas ferramentas específicas do sistema NMSIS. Surge neste capítulo, a necessidade de conhecer o estado da arte actual, dos princípios que regem a correlação de eventos.

3.2. CONCEITO DE CORRELAÇÃO

Em teoria da probabilidade e estatística, a correlação, também chamada de coeficiente de correlação, indica a força e a direcção do relacionamento linear entre duas variáveis aleatórias. No uso estatístico geral, a correlação ou co-relação refere-se à medida da

relação entre duas variáveis, embora correlação não implique causalidade. No sentido geral, existem vários coeficientes que medem o grau de correlação, adaptados à natureza dos dados. Interessa neste trabalho fazer um estudo introdutório à correlação de eventos, no que consistem e como podem ser exequíveis na prática.

3.3. UTILIZAÇÃO DA CORRELAÇÃO NOS EVENTOS DE REDE

A correlação de eventos é a funcionalidade chave de um sistema de gestão de redes que permite filtrar os eventos redundantes e artificiais de modo a determinar a raiz que causa as falhas de uma rede. Um sistema de correlação normalmente combina modelos de correlação causais e temporais com a topologia da rede [122]. Cada sistema tem o seu modelo e o seu próprio algoritmo e a sua robustez é diferente de sistema para sistema.

O paradigma que um sistema de correlação tem que realizar e tentar solucionar, são as linhas principais e os alicerces de concessão deste projecto de trabalho.

Quais são os principais objectivos de um sistema de correlação [122]:

1. Diagnosticar a raiz do problema que causa falhas na rede e consequentemente degradação na performance, estabelecendo relações entre eventos da rede.
2. Filtragem de eventos (Alarmes de rede) que inundam a rede correlacionando-os num único evento conceptual.

Para que tenha utilidade prática, o sistema deve ser:

1. Correcto: A raiz da causa inferida pelo sistema deve ser com alta probabilidade a implicação fundamental para todos os eventos que foram detectados. Ou seja, a raiz da causa deve ter verdadeiramente ocorrido na rede.
2. Ideal: O sistema deve inferir um pequeno número de raízes das causas que podem explicar todos os eventos detectados.

Devido à importância de um sistema de correlação, já vários sistemas [104][105][106][107][108] foram propostos e implementados. Uma comparação entre estes vários sistemas pode ser encontrado no [103]. Nesta diversidade de sistemas de correlação, em que cada um tem a sua própria descrição e implementação, será difícil a sua comparação [122]. Na ausência de uma definição simples do problema que se pretende definir através da correlação de evento e de um sistema independente do software usado, é impossível analisar um sistema correcto e ideal proposto para o futuro.

Neste trabalho pretende-se estudar o problema da correlação de eventos, por isso terá que ser definido as relações de correlação que serão correctas de inferir e depois chegar à conclusão que o sistema foi escolhido devido a determinado número de razões. Para isso, devemos:

1. **Primeiro**: Tirar algumas intuições acerca de eventos e falhas;
2. **Segundo**: Tirar algumas intuições acerca de correlação de eventos;
3. **Terceiro**: Estas intuições motivam o desenvolvimento de correlação causal;
4. **Quarto**: Estas intuições motivam o desenvolvimento de correlação temporal;
5. **Quinto**: Ilustrar como o novo sistema ou ferramenta a ser apresentado, poderá ser útil se usado na rede proposta.

3.4. EVENTOS E FALHAS

O sistema deverá correlacionar eventos para diagnosticar as falhas. O evento é uma ocorrência instantânea a determinado momento no tempo. Para propósitos desta tese é suficiente dizer que tipicamente o evento está associado a um objecto no qual ele ocorre. O objecto é uma entidade com determinado estado, para modelizar o estado de um objecto, o objecto deve ser tratado como uma colecção de parâmetros cujos valores podem mudar com o tempo e com esses parâmetros ligados ao estado do objecto a cada instante temporal.

A falha é um evento que está associado a um estado anormal da rede, isto é, comportamento que se desvia do normal e esperado [122]. Este desvio pode ter inúmeras razões, falhas de *hardware* ou *software*, erros humanos, erros de desenho de rede, ou uma

combinação de várias falhas. Eventos podem ser nativos ou compostos, são nativos se forem gerados internamente ou corresponderem a uma alteração do estado do objecto, porque são directamente observáveis. Por outro lado podem ser compostos e não são directamente observáveis e não podem ser detectados por *hardware* ou *software*, porque o objecto onde os eventos ocorrem não é monitorizado, ou o evento não é reportado. Por exemplo, uma falha de uma ligação de um *router*, pode não ser directamente reportada.

A ocorrência de um evento não observável é estabelecida através de um padrão de ocorrência de outros eventos ao qual o evento não observável está correlacionado. Por exemplo, a falha de uma ligação de um *router* pode ser inferida através da falha até um servidor e um cliente ligado ao *router*. Outros eventos no qual o evento não observável está correlacionado podem eles próprios, não serem observáveis também [122].

3.5. CORRELAÇÃO DE EVENTOS

A correlação de eventos trabalha sobre o principio de estabelecer uma relação entre eventos de rede, determinado evento "a" correlaciona uma serie de eventos a1, a2, …, ak, de forma que "a" => {a1, a2, …, ak}, se a1, a2, …, ak entrarem numa relação entre todos e com "a" definirem um padrão [122].

- A detecção e isolamento de falhas, é um passo importante para descoberta da causa mais provável, que está na raiz do problema. Se tivermos uma simples rede constituída por clientes que estão ligados servidores de serviços através uma rede de *routers* e *switchs*.

  Nesta rede monitorizada, os alarmes que surgem de falhas no servidor " *falha na conexão ao Servidor"* ($s_¡$), ou falhas que surgem de clientes de serviços " *falha na conexão ao serviço"* ($c_¡$), podem ser atribuídas a falhas nos *routers* " *falha na ligação do router"* ($r_ı$), ou serem atribuídas a falha nos *switchs* " *falha na ligação do switch"* ($s_ı$). Sem este conhecimento que $r_ı$ ou $s_ı$ => $s_¡$ ou $c_¡$, um operador de rede ou um sistema automatizado não sabe apontar a raiz da causa do problema de uma forma efectiva e eficiente, numa rede grande e com muitos alarmes de "falha de conexão" [122].

- Filtragem e supressão de eventos, é outra característica importante que o sistema deve ter. Por exemplo numa rede simples o sistema de correlação deverá reportar apenas

alarmes primários, tais como estados de *switchs* ou *routers* (r1, s1), enquanto suprime os eventos secundários e potencialmente numerosos de falhas do tipo "*falha na conexão"*.

- Ajuste de performance, é conseguido quando a raiz do problema das falhas de rede é isolada eficientemente, quando a degradação da rede dura um tempo mínimo e quando os objectos passíveis de monitorização e relacionados com as falhas de rede podem ser ajustados de forma efectiva.

As relações entre eventos usados para correlação e filtragem podem ser classificados como sendo causais ou temporais [122]. Será visto nos próximos capítulos estas duas formas de correlação e iremos ilustrá-las com exemplos as relações que podem ser combinadas em conjunto, para acompreensão da correlação de eventos.

3.6. CORRELAÇÃO CAUSAL E FILTRAGEM

A relação mais simples entre eventos que pode ser considerada é a relação causa - efeito ou de causalidade. A maioria das fontes de alarme das redes são as falhas físicas que ocorrem nos elementos de rede. Estas falhas podem ter uma relação de causalidade ou não (isto é, podem ser independentes).

As relações de causalidade entre as falhas podem ser representadas por um gráfico (figura 10) de propagação de falhas. Todas as falhas que não estão directamente exibidas por alarmes podem ser reconhecidas correlacionando múltiplos alarmes. Por exemplo, a correlação dos alarmes "a2" e "a3" (correlação c2) permite que um possa fazer a decisão de diagnóstico, que a raiz da causa destes alarmes é a falha "f4". No entanto, a existência das duas correlações "c1" e "c2" leva ao reconhecimento da falha "f2", como a raiz causal das falhas. Uma falha pode ser causada pela presença de apenas uma ou várias falhas [90].

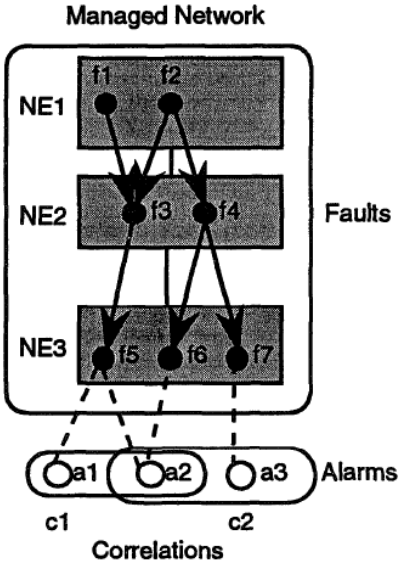


Figura 10 Gráfico de propagação das falhas [90].

A correlação causal e filtragem, é a resposta para chegar à raiz do problema. Determinado evento "a" é causa do evento "a' ", ou seja a → a'. Desta forma pode ser desenhado um esqueleto ou árvore de causa - efeito, como o representado na figura 11 [122]. Este gráfico ilustra de uma forma directa e não cíclica, a representação de eventos através dos nós "○" e a representação de relações entre os nós, através das setas "→". É também conceptualmente mais fácil definir a correlação e filtragem através do gráfico. Esta figura exemplo mostra uma estrutura de causa – efeito, com um conjunto de eventos {1, …, 13}. Dada a informação causal do gráfico sobre os eventos, um dos objectivos do trabalho é deduzir as relações de correlação "a" => {a1, a2, …, ak}, ou seja, o evento "a" correlaciona um determinado conjunto de eventos reportados a1, a2, …, ak. Se todos os eventos a1, a2, …, ak forem reportados, o sistema pode ser usado para filtrar (suprimir) esses eventos com a representação de um único evento representado por "a".

Considerando as relações de causalidade no exemplo da figura 11. Será razoável pensar que podemos utilizar o evento "8" para correlacionar os eventos {2, 3, 6}, já que o evento "8" causa os eventos 2, 3, 6, ou seja, "8" => {2, 3, 6}. Também se pode concluir que "8" => {1, 2, 3, 6} já que o evento "8" causa o evento "1" transitivamente através de "2". Por outro lado, não queremos concluir que o evento "8" correlaciona o conjunto {2, 3, 6, 7}, já que o evento "7" não poderá ser causado pelo evento "8". Outra coisa que não se pretende definir é que "8" correlaciona o conjunto {3, 6}, embora "8" seja a causa de ambos os

eventos “3” e “ 6”, será incorrecto concluir a presença de “8” como a causa. Uma vez que, de facto o evento “8” ocorreu, em seguida, teríamos também de ter visto o evento “2,” que no entanto não é relatado no conjunto. Certo tipo de eventos podem ser não-observáveis, como tal, não exige necessariamente a presença explícita de todas as consequências provocadas por um evento no conjunto de eventos relatados. Por exemplo, “8” => {1,3,6}, uma vez que mesmo que o evento “2” não esteja presente no conjunto de eventos relatados, podemos inferir a sua ocorrência através da notificação do evento 1.

Definir uma relação de correlação entre eventos nem sempre é simples, por essa razão será demonstrado mais a frente os exemplos reais propostos para matéria de estudo deste trabalho.

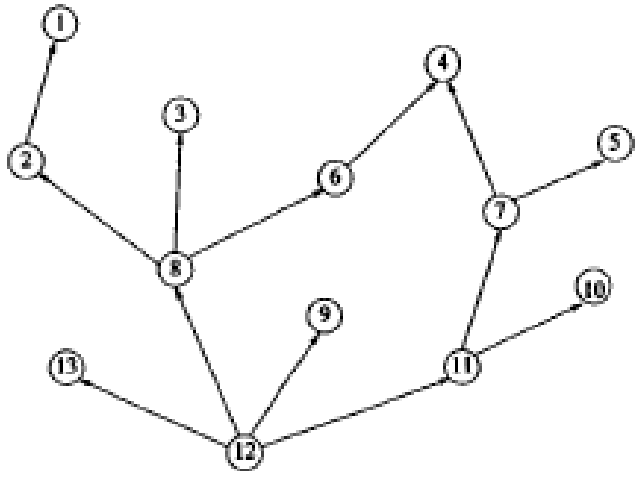


Figura 11 Exemplo de uma árvore ou gráfico de causalidade [122].

3.7. CORRELAÇÃO TEMPORAL E FILTRAGEM

Podemos estabelecer correlações mais ricas entre eventos com base num determinado instante temporal da sua ocorrência. Para isso é preciso definir relações temporais entre os eventos. Alguns exemplos de relações temporais são:

a) Evento seguido de outro evento ;

b) Primeiro evento desde o recente evento ;

c) Evento seguido de evento dentro de uma janela temporal de τ, ou não (o evento não foi observado no intervalo de tempo τ).

Conceptualmente os eventos causais anteriormente descritos são obtidos através da interpretação dos eventos como se de proposições que tem os valores de forma booleana em verdadeiro ou falso (1 ou 0) se tratassem, indicando a sua ocorrência [122]. Para representar a relação temporal entre eventos é preciso definir a interpretação dos eventos para incluir informação acerca do instante temporal a que cada evento ocorreu. O tempo pode ser definido como um conjunto de τ com uma ordem linear. Podemos interpretar um evento como sendo função do tempo τ com os estados {verdadeiro, falso} e com instantes de ocorrência definidos pelos argumentos (instantes temporais) no qual ele é verdadeiro.

3.8. CLASSIFICAÇÃO DAS OPERAÇÕES DE CORRELAÇÃO

Existem vários tipos de operações que podem ser executadas nos eventos, durante o processo de correlação. Vários autores abordaram esse tema [90], **Error! Reference source not found.**, [125], estas operações podem ser classificadas da seguinte forma [123]:

- **Compressão** – Substitui repetidos eventos A, por um único.
- **Supressão** – Suprime um evento A, se determinado contexto operacional estiver presente.
- **Filtragem** – Suprime um evento A, se um dos seus parâmetros possuir determinado valor.
- **Contagem** – Conta eventos repetidos A, se o seu número exceder determinado limiar (*treshold*), substitui-os por um único evento B.
- **Dimensionamento** – Na presença de certo tipo de contexto operacional, substitui um evento A, por um evento B, quando determinado valor de um parâmetro tem um valor acima do estipulado.
- **Generalização** – Substitui um evento A por um evento mais genérico B, se determinado contexto operacional estiver presente.
- **Especialização** – Substitui um evento A, por um evento mais especifico B, se determinado contexto operacional estiver presente.

- **Relação Temporal** – Correlaciona eventos dependendo da ordem de chegada e/ou tempo da sua geração.

- **Clustering** – Gera um evento A, se a amostras (patterns) de correlação mais complexas forem detectadas nos eventos recebidos. A operação de clustering pode ter em atenção o resultado de determinados testes externos, ou a combinação de várias operações de correlação anteriores.

3.9. TÉCNICAS DE CORRELAÇÃO DE EVENTOS

Um número elevado de métodos foi adoptado na investigação em inteligência artificial para realizar operações de correlação. Estas aproximações podem ter várias categorias distintas [161]. Nas técnicas de correlação de eventos é sempre necessário ter algumas premissas que servem de base ao raciocínio e que permitem chegar a determinado tipo de conclusões, dado que os sistemas só conseguem processar e aferir problemas com determinada complexidade. A correlação de eventos pode ser obtida através de vários tipos de aproximações, ou mesmo através da sua combinação. O autor [160], fala da comparação destas técnicas, em termos de manutenção, modelização, robustez e performance do sistema de correlação. Algumas dessas técnicas mais conhecidas para correlação de eventos, são:

- *Rule-based reasoning* (RBR);

- *Model-based reasoning* (MBR);

- *Case-based reasoning* (CBR);

- *Code-base (codebook) correlation* (CC);

- *State Transtion base* (STB);

- *Probabilistic Finite State Machine (PFSM);*

- *Network Element Dependency Graph (NEDG);*

- *Hybrid approach.*

3.9.1. RULE-BASED REASONING

Os sistemas do tipo *Rule-based reasoning* dependem da colecção de regras para permitir fazer as suas decisões. Cada uma das regras é definida com uma série de condições que devem ser satisfeitas antes de determinada accção acontecer. Se forem satisfeitas todas as condições numa regra, então a respectiva acção terá lugar. O sistema que faz a verificação é conhecido como motor de inferência. A secção de memória utilizada para salvar informação enquanto os processos verificação ocorrem é conhecida como memória de execução.

Se Condição_A E Condição_B THEN Acção_1

Figura 12 Exemplo de uma regra para um sistema rule-based reasoning.

Existem também versões especializadas de aproximação com modelo *rule-based reasoning* para correlação de eventos. A operação básica é similar aos dos sistemas de *rule-based*, com as mesmas características e módulos necessários. No entanto, o sistema especializado faz uso do conhecimento especializado deliniado nas regras de base para guiar os seus processos de decisão. Produzem resultados de alta qualidade, verdadeiros e de confiança. Estes sistemas são aplicados em casos em que as condições não mudam, e o conhecimento de todo o núcleo central é tudo que é necessário [163].

3.9.2. MODEL-BASED REASONING

Em inteligência artificial, o modelo "*model-based reasoning*", refere-se a um método de inferência, usado nos sistemas especializados, e que têm por base o modelo do mundo físico real. Na prática este modelo faz o estudo de experências e factos (observações, dados, eventos, fenómenos), reduzindo isso a uma determidada ordem científica ou padrão (gráficos, generalizações, leis) e depois converte numa inquirição científica ou explicação (modelos, hipóteses, teorias).

No sistema com o modelo "*model-based reasoning*", o conhecimento é representado com regras causais [156]. O método de aproximação *model-based reasoning*, procura uma forma de criar um duplicado de software do sistema que está ser monitorizado. Quando um

evento ocorre, o modelo de software procura o rasto do evento para trás até a sua fonte para determinar a causa [163]. Quando a causa é determinada o sistema reage de acordo com as suas definições.

3.9.3. CASE-BASED REASONING

A aproximação *case-base reasoning* é de todas as opções disponíveis, a mais complexa para correlação de eventos [161]. Este método faz uso do armazenamento de experiências passadas, que modifica para safisfazer a situação currente. Num sistema de *case-based* cada evento que chega é avaliado, a informação extraída durante este processo de avaliação é usada para recuperar experiência relacionadas. Estas experiências são depois modificadas para satisfazer a nova situação descrita no evento. Uma vez a situação tratada, toda a informação referente é armazenada como experiência para uso futuro. Com algoritmos correctos, o método de raciocínio *case-based* pode aprender como responder eficazmente a situações no seu ambiente [161].

3.9.4. CODE-BASE CORRELATION

Na aproximação *code-based reasoning* para correlação de eventos, as regras para responder a um evento são representadas numa matriz. Para matrizes extremamente grandes, algoritmos de compressão podem ser aplicados, para produzir uma versão compacta com tamanho de armazenamento e tempos de pesquisa reduzidos. A miniaturizada matriz resultante é conhecida por "code", daí o nome para a aproximação.

	A1	A2
E1	1	0
E2	0	1
E3	0	1

Figura 13 Matriz de correlação de um sistema code-base reasoning.

A matriz de correlação consiste em eventos e acções. A existência do valor "1" numa coluna, demonstra a necessidade do evento correspondente, ocorrer antes do curso da acção representado pela coluna. Por exemplo, a acção A1 só pode ocorrer após o evento E1 ter ocorrido, ou seja, existe o valor de "1" na célula {A1, E1}. A acção A2 só pode ocorrer após o evento E2 e E3 ter ocorrido, ou seja, existe o valor de "1" nas células {A2, E2} e {A3, E3}. A ocorrência de E2 e E3 independentes não faz disparar a acção correspondente,

apenas a ocorrência em conjunto. Este tipo de matrizes e *code-books* fornecem ganhos significativos [163].

3.9.5. STATE TRANSTION BASE

Os sistemas com aproximação do tipo *State Transtion base* fazem uso de três componentes: *token*, *state* e *arc* [163]. Os arcos ligam diferentes estados e agem como se fossem um caminho para o *token*, permitindo que se movam entre os estados. Quando um *token* entra em determinado estado uma acção é executada. Essa acção pode ser para notificar o operador de um possível problema. O movimento do *token* é accionado pela chegada de eventos. Se a estrutura for especializada num tipo específico de eventos e acções, é necessário construir gráficos de transição separados para cada tipo de de componente no sensor da rede. Após a implementação completa o sistema é muito similar ao *model-based reasoning*, a quando da criação de um modelo, embora não de todo o sistema, simplesmente as possíveis interacções entre componentes.

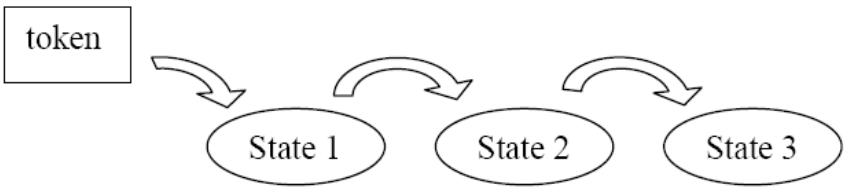


Figura 14 Estrutura de um gráfico de transição [162].

3.9.1. PROBABILISTIC FINITE STATE MACHINE

A aproximação de correlação de eventos proposta nos modelos do autor [164], em que cada falha usa a *Probabilistic Finite State Machine* (PFSM). Esta aproximação tem a particularidade de manipular sequências de eventos com ruído e utiliza teoria probabilística para selecionar o PFSM correcto na sequência de falhas de entrada. O processo de construção do PFSM para cada falha é um processo de auto-aprendizagem que usa a teoria probabilística. E a associação de informações de falha para o PFSM, podem ser obtidas através de outros sistemas ou redes especializados. Esse algoritmo não assume qualquer conhecimento da estrutura de rede e pode reconhecer automaticamente o tempo padrão de alarmes associados a uma determinada falha.

3.9.2. NETWORK ELEMENT DEPENDENCY GRAPH

Inúmeras aproximações de correlação são construídas com base em *Network Element Dependency Graph* (NEDG) [165], [166]. Estas aproximações à base de gráficos de dependência de elementos de rede, são usados para modelizar a dependência funcional entre os elementos na rede de gestão de objectos. Este tipo de abordagem permite localizar as falhas com grande facilidade e em seguida usar o gráfico de dependência de elementos de rede, para realizar a correlação. No entanto, precisa de transportar explicitamente, informações de localização nas mensagens de erro, através do uso dos protocolos de gestão, como por exemplo o SNMP, mas como o SNMP não oferece este tipo de suporte, por isso, limita a ampla utilização desta abordagem.

No algoritmo positivo proposto em [165], um algoritmo de correlação de alarmes usa gráficos de dependência de rede. Os alarmes emitidos pelo objecto de rede serão explicitamente associados com informações sobre a localização para indicar o componente com falha ou com mau funcionamento. A correlação será realizada sobre o gráfico de dependência do elemento rede, usando as informações associadas com cada alarme recebido.

3.9.3. HYBRID APPROACH

Algumas abordagens híbridas também têm sido propostas para unificar múltiplos tipos de abordagens em conjunto, para que essas abordagens complementares possam trazer novos benefícios quando combinadas. O trabalho [167], propõe uma correlação de eventos híbrido combinando *rule-base reasoning* e a abordagem *code-based reasoning*. O raciocínio baseado em regras é usado na correlação de nível baixo e é normalmente usado para correlacionar os eventos que ocorrem dentro de um dispositivo de rede. A abordagem *code-based*, porém, é usada para correlacionar eventos de redes diferentes. E é geralmente usada para correlação de alto nível. Esta abordagem adopta diferentes técnicas de correlação para diferentes níveis de correlação.

Concluído o estudo dos diferentes métodos para inferir relações entre eventos. Os tipos de operações que podem ser realizadas durante o processo de correlação de eventos. E as técnicas que servem de base ao raciocínio durante os processos de correlação de eventos. Posto isto, é possível entrar no estudo das ferramentas de correlação de eventos que actualmente existem no mercado ou em regime de código fonte aberto.

3.10. FERRAMENTAS COM CORRELAÇÃO DE ALARMES

A correlação de eventos tornou-se uma das técnicas mais importantes nas redes de gestão. O seu principal objectivo é permitir a redução de eventos de rede para um menor número e mais significativo no conjunto de mensagens de alarmes que podem ser manipulados por um operador, numa janela temporal mais conveniente.

Neste contexto, será feito uma pesquisa a um leque alargado de ferramentas que tem como principal objectivo o processamento e correlação de eventos. Neste sentido, a orientação deste trabalho de Tese de Mestrado será na procura e estudo de ferramentas de código fonte aberto, segundo os termos do *Free Software Foundation* (FSF) e mais especificamente, ferramentas do sistema operacional *General Public License* (GNU). Ou seja, software completamente livre e grátis sem qualquer tipo de restrições. No entanto, serão abordadas algumas ferramentas comerciais, para dar conhecimento dos fabricantes que existem no mercado com este tipo de tecnologia, mas não terá qualquer aplicação prática para os objectivos do trabalho proposto. Serão por isso estudadas, nove ferramentas de correlação de eventos: *Snort, SnortSnarf, SnortCenter, HP OPenView ECS, CLIPS, IMPACT, Swatch, LogSurfer e SEC,* para se escolher qual delas será utilizada num sistema integrado com o NMSIS.

3.11. SNORT

Originalmente lançado em 1998 por Martin Roesch, fundador da *SOURCEfire*. O Snort é uma ferramenta *Network Intrusion Detection System* (NIDS) bastante popular, combina os beneficios das assinaturas, protocolos e inspecção de anomalias. É um sistema Open Source de detecção e prevenção de intrusão, capaz de de fazer análise de tráfego em tempo real e fazer *logs* de pacotes em redes IP.

Inicialmente conhecido por um sistema leve de detecção de intrusão “lightweight”, o Snort evoluiu para um sistema maduro, que faz com que actualmente seja dos mais conhecidos e utilizados sistemas de intrusão e detecção [129]. O código fonte é optimizado, é desenvolvido em módulos, que utilizam linguagem de programação C e, juntamente com a documentação, são do domínio público. A configuração é simples e aplicada através de um sistema de arquivos organizado, documentado e de fácil manutenção.

Características e Funcionalidades Especificas:

- Não Suporta Correlação, só quando utilizado com outras ferramentas; Ferramenta *Open Source*; Análise de Protocolos; Nas consultas dispõem de mecanismos de visualização e de análise dos dados; Procura de matching do conteúdo dos protocolos; Usado para vários tipos de ataques (*buffer overflows*, *stealth port scans*, *CGI attacks*, *SMB probes*, *OS fingerprinting attempts*, etc); Sistema de regras (linguagem por regras flexivel) para saber o que colectar; Motor de detecção que utiliza uma arquitectura modular (em pug-in); Capacidade de alerta em tempo real; Faz armazenamento em ficheiros (*logs*); Tem sistema de alerta, por socket Unix ou por mensagens Winpopup para clientes Windows; Boa Documentação; Boa portabilidade.

Requisitos de sistema e licensiamento:

- Os requisitos para correr o *Snort*, ter em Unix o seguinte software instalado *Libpcap*, PCRE, Libnet, Barnyard. Se correr em Window deve ter apenas o winpcap instalado. O licensiamento do código fonte do *Snort Engine* e das regras da comunidade *Snort Rules* é da *General Public License Version 2* (GNU).

Observações Gerais:

O SNORT é uma ferramenta NIDS open-source bastante popular pela sua flexibilidade nas configurações de regras e constante actualização. Outro ponto forte desta ferramenta é o facto ser leve, pequeno, de fácil instalação e de ter o maior cadastro de assinaturas face aos concorrentes (NIDS). Permite verificar todo e qulquer tipo de anomalias dentro da rede no qual os computadores pertencem. O Snort monitoriza o tráfego de pacotes de redes IP, realizando análises em tempo real sobre diversos protocolos.

Outro ponto positivo desse software é o grande número de possibilidades de tratamento dos alertas gerados, O subsistema de registro e alerta é selecionado em tempo de execução através de argumentos na linha de comando, são três opções de registro e cinco de alerta [129]. Os alertas podem ser enviados ao *syslog* e registrados num arquivo de texto puro em dois formatos diferentes, ou podem ser enviados como mensagens *WinPopup* usando o "*smbclient"*. Os alertas podem ser enviados para arquivo texto de forma completa. O Snort é uma ferramenta de detecção de invasão rápida e confiável que exige poucos recursos de sistema. Por ser uma ferramenta peso leve, a utilização do Snort é indicada para

monitorarizar redes TCP/IP de qualquer tamanho dependendo da funcionalidade que será exigida, onde pode detectar uma grande variedade do tráfego suspeito, assim como ataques externos, para fornecer informações de tomada de decisão aos administradores [130].

O Snort pode trabalhar em *stand alone*, mas existem várias pacotes com ferramentas muito uteis (*add-ons*) para serem usados com o Snort de uma form mais facil e flexivel. O Snort só por si não suporta correlação, mas quando integrado com outras ferramentas, isso já é possível. Podemos ver exemplos de algumas dessas ferramentas que quando combinadas com o Snort permitem fazer acções de correlação.

3.12. SNORTSNARF

O SnortSnarf é um programa em *Perl* que pega nos ficheiros de alertas do *Snort Intrusion Detection System*, e produz uma saída em *HTML* destinada a inspecção de diagnóstico e rastreio de problemas [133], [134]. As páginas HTML facilitam a visualização dos alertas gerados pelo Snort. Executado periodicamente recebe como argumentos os arquivos "*alert*" e "*portscan*".

Características e Funcionalidades Especificas:

- Suporta Correlação limitada; Ferramenta *Open Source*; Faz filtragem de eventos; Entrada são os ficheiros de alerta do Snort no directório "*\lo"* ou a base de dados do Snort; Saída em Web (*HTML)*; Inspecção de diagnóstico e rastreio de problemas; Utiliza o sistema de regras do Snort; Não permite alerta em tempo real; Não tem Documentação; Boa portabilidade.

Requisitos de sistema e licenciamento:

- Os requisitos para correr o *Snort*Snart são semelhantes aos do Snort. O licensiamento do código fonte do *SnortSnarf* é licenciado segundo os termos do *General Public License* (GNU) e publicado pela *Free Software Foundation*. Que dá permissões de copiar, distribuir e/ou modificar o *SnortSnarf* dentro de certas condições.

Observações Gerais:

O SnortSnarf pode remeter para a sua entrada, os ficheiros de alerta do Snort no directório "*log*" ou da base de dados do Snort, para criar os ficheiros de saída em HTML. A consola "*Snortsnarf console"* não permite a visualização de alertas em tempo real [145]. A utilização do *SnortSnarf* facilita bastante a visualização dos alertas por produzir uma boa interface, onde o administrador pode utilizando recursos de *browsers* comuns, navegar pelos alertas e mais rapidamente identificar problemas mais graves.

Possui algumas opções, tais como: a) **d** <diretório> especifica o directório onde serão gerados os arquivos HTML, esse diretório deve estar num caminho dentro do diretório base do servidor *web* para poder ser acedido através de um browser comum; b) **ldir** <URL> - dá a possibilidade de criar *hiperlinks* nas páginas geradas para os arquivos com o conteúdo dos pacotes capturados. Essa opção especifica a *Uniform Resource Locator* (URL) para o diretório onde estão contidos os registos; c) ***rulebase*** <arquivo de regras> - Especifica o arquivo de regras que é usado pelo Snort. Muito útil para incluir nas páginas geradas, *links* para descrições dos ataques detectados.

Uma desvantagem é a utilização adicional de recursos dos sistemas onde estão os sensores e o defasamento na visualização dos alertas, principalmente em janelas de alto tráfego. Em experiências realizadas, chegou-se a conclusão que sua utilização deve ser associada a outros métodos de visualização tradicionais (exame manual dos registos) aconselhando-se a sua execução em horários de pouco movimento. Outro problema encontrado foi a incompatibilidade inicial da visualização via web dos logs com o esquema de rotação e de compactação de logs. Quando os logs eram rotatórios e compactados deixavam de ser visualizados pela página.

3.13. SNORTCENTER

O SnortCenter é um sistema de gestão do tipo cliente-servidor, feito para ser utilizado em *Web*, é escrito em PHP e Perl [146]. O SnortCenter ajuda a configurar o Snort e a manter as assinaturas actualizadas. A consola de gestão constroi os arquivos de configuração e envia aos sensores remotos. Tanto a consola de gestão, como os agentes podem ser instalados em sistemas Unix ou Windows. Permite a encriptação do tráfego cliente-servidor, a autenticação e updates automáticos de assinaturas.

Características e Funcionalidades Especificas:

- Suporta Correlação limitada; Ferramenta *Open Source*; Faz filtragem de eventos; Saída em Web (*HTML)*; Encriptação SSL entre o sistema de gestão e os sensores agentes; *Update* automático que importa novas assinaturas Snort da Internet e coloca-as nos sensores; Faz o *Start-Stop* do Snort remotamente coloca a configuração específica no sensor; Cria regras pessoais e modifica as regras existentes; Sistema que suporta *Templates* de regras para configurar mais facilmente múltiplos sensores; Um "Sensor Agente" pode lidar com multiplos *daemons* Snort, se o sistema tiver múltiplas interfaces de rede; Consola de gestão e *Sensor Agents* para Linux, BSD, *NIX, Windows; Não permite alerta em tempo real; Tem boa Documentação; Boa portabilidade.

Requisitos de sistema e licenciamento:

- Os requisitos para correr o *Snort*Snart são semelhantes aos do Snort. A consola de gestão: Webserver (apache); PHP (compilado com with-mysql); MySQL; URL *command line tool* (com suport SSL). O agente Sensor (*Sensor Agent*), remoto ou na consola de gestão: Snort; Perl 5. O licensiamento do código fonte do *SnortCenter* é licenciado segundo os termos do *General Public License* (GNU) e publicado pela *Free Software Foundation*.

Observações Gerais:

O SnortCenter faz a gestão dos sensores remotos com base num ambiente Web, seguindo o método cliente-servidor. A função de filtragem de eventos no SnortCenter é rápida e eficiente. O ponto-chave é lembrar que todos os sensores partilham o mesmo ficheiro "*local.rules"*. A forma como se controla quais os sensores que são afectados pelo filtro, é através da opcção "*SENSOR SCOPE"*. Com o SnortCenter e um sistema IDS como o Snort e com um ambiente que usa *Network Address Translation* (NAT), é possível, através de correlação de eventos entre o sistema IDS, antes e depois da firewall, tirar o endereço real da fonte que fez a tentativa de entrar em determinada rede. Os sensores têm um conjunto de características: o propósito destas características é documentar e entender o tráfego que atravessa a ligação onde o sensor está localizado.

Pode-se usar esta informação para cortar falsos positivos, para ajustar os sensores e para encontrar anomalias no tráfego. Existe um formato específico para populares estes campos e uma das características, é a relação com outros sensores. Se por exemplo um sensor antes e depois do *proxy*. Se se vir um alerta no sistema IDS depois do *proxy* e queremos o endereço da fonte real, é preciso referenciar o sensor antes do *proxy*.

3.14. HP OPENVIEW ECS

O HP OpenView *Event Correlation Services* (ECS) é uma tecnologia avançada de correlação de eventos, que fornece um dos mais inteligentes motores de correlação actuais. O ECS é totalmente integrado com o nó gestor de rede, *Network Node Ma*nager (NNM) para correlacionar eventos de rede e operações *HP OpenView*. E também, para correlacionar eventos que vêm de diferentes camadas, tais como: rede, sistemas, aplicações, bases de dados e Internet. Este produto dá aos seus clientes uma forma de resolver muitos dos seus problemas. Quando um problema ocorre, o nó gestor de rede e de operação filtrará os eventos, e irá correlacionar esses eventos e reportar aos administradores a real causa do problema [130].

O modelo ECS é um produto bastante poderoso com capacidades de criar regras específicas para correlação de eventos que podem ser usados numa rede com nó gestor de rede e de operação. O tipo de ECS *Designer GUI*, torna o complexo processo de correlação de eventos de forma intuitiva, mais amigo do utilizador. As regras de correlação são criadas utilizando um paradigma visual de nós ligados entre si para formar um circuito de correlação [107].

Características e Funcionalidades Especificas:

- Suporta Correlação; Ferramenta Comercial; Definem-se "circuitos de correlação" para o processamento de eventos; Correlação de eventos em tempo real; Permite criar correlações simples através da interligação das primitivas de um nó; Sistema de regras; Supressão de informação; Filtragem de eventos; Modelo de execução por "*rule-based reasoning*"; Permite testar as regras definidas em *simulate mode*; Tem tecnicas para fazer *feedback* dos dados do circuito de correlação; Capacidade de alerta em tempo real; Utiliza armazenamento de eventos em ficheiros (*logs*); Boa Documentação (somente a quem adquire o produto).

Requisitos de Sistema e licenciamento:

- Existem requsitos para o software e hardware no *network node manager 6.x* e no *hp OpenView operations 5.x* [130] Este produto é agora integrado com as versões correntes do *Network Node Manager and Operations* (NNMO). O licensiamento do produto e a sua encomenda é apenas requersitado para o *Network Node Manager* (NNM 6.1) e anteriores, e *Operations 6.x* e anteriores. O motor *ECS Engine*, não tem custos com qualquer produto nó gestor de rede e de operação *Network Node Manager and Operations* (NNMO), permitindo correlação *out-ofthe-box.*

Observações Gerais:

O *Network Node Manager* (NNM) da família *OpenView* da *HP*, é das plataformas mais divulgadas e utilizadas no contexto de gestão de redes. O NNM possui, como uma das suas maiores vantagens, uma componente gráfica que se organiza em diferentes símbolos e cores, que possibilita através de uma inspecção visual identificar problemas na rede. Uma outra aposta por parte da *HP* é a disponibilização de uma interface *WEB* baseada em *JAVA* para as suas aplicações de gestão, o que possibilita o acesso a partir de qualquer ponto da rede à aplicação de gestão.

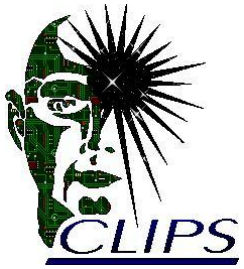
Na plataforma *OpenView*, existe o *Event Correlation Services* (ECS). No ECS programam-se "circuitos de correlação" para definir o processamento de eventos [82]. Internamente o ECS usa um tipo de linguagem de programação funcional. Com a utilização do paradigma visual de nós ligados entre si para formar um circuito de correlação [107]. Os eventos entram nos nós, por via das portas de entrada e saem pelas portas de saída. Uma porta de saída de um nó está ligada a uma porta de entrada de outro nó, dentro do circuito, para que os eventos fluam através do circuito de um nó até ao próximo. Os tipos de nós fornecidos pelo ECS são um conjunto de tipos básicos, considerados necessários e suficientes para fazer correlação de eventos em tempo real.

Existe outra aproximação implementada para a área de telecomunicações que está presente no *HP OpenView Fault Management Platform* (FMP), que é parte integrante do *HP OpenView Element Management Framework* (OEMF). Neste, o utilizador cria regras que descrevem a relação causa-efeito. Esta abordagem pode ser combinada com o próprio ECS. No ECS os eventos são analisados à medida que chegam ao "*Alarm Browser"* e vão sendo relacionados, de forma a encontrarem a causa do problema, por exemplo quando chegam

vários eventos do tipo "*interface down"* do mesmo dispositivo e depois chega um evento do tipo "*node down*" desse mesmo dispositivo, apenas o último é colocado no "*Alarm Browser"*, enquanto que os outros são correlacionados a este. Existem várias formas, previamente definidas, para correlacionar eventos, mas também é possível modificá-las e adicionar novas estratégias.

Nesta ferramenta, a componente do subsistema de eventos, transforma e processa o fluxo de eventos de acordo com as correlações instaladas. O ECS dá a flexibilidade de correlacionar mensagens de vários tipos, tais como: SNMP, ASCII e eventos de qualquer fonte de mensagens de operação. Estes eventos são processados e visualizados num unico e consistente formato, independentemente do tipo de evento [130].

3.15. CLIPS



O CLIPS é uma ferramenta especializada, que oferece um ambiente completo de desenvolvimento e de produção para a construção e execução de sistemas à base de regras e/ou à base de objectos [89]. Foi criado em 1995, no *Lyndon B. Johnson Space Center* da NASA. Os fundos acabaram nos princípios dos anos 1990, e houve um mandato da NASA para comprar software comercial. O *C Language Integrated Production System* (CLIPS) é actualmente muito usado nos governos, industrias e academicamente. O CLIPS é um acrónimo de Sistema de Produção Integrado em Linguagem C. Na actualidade, entre os paradigmas de programação que suporta CLIPS, encontram-se: a Programação lógica; a Programação imperativa; a Programação Orientada a Objectos. O CLIPS provavelmente é o sistema para especialistas, mais dessiminado devido a um conjunto de características: é rápido, eficiente e gratuito. Mesmo sendo do domínio público, é ainda actualizado e mantido pelo seu autor original, *Gary Riley*.

Características e Funcionalidades Especificas:

- Suporta Correlação; Ferramenta em *Open Source*; Modelo de execução por inferência; Modelo de execução por "*rule-based reasoning*"; Não permite correlação de eventos em tempo real; Sistema de regras; Geração de programas que modelizam o conhecimento ou alguma especialização humana (sistemas especialistas); Plataforma de desenvolvimento e produção para sistema de regras e/ou objectos; Facil de integrar com outras linguagens (C, Java, FORTRAN and ADA); Facil de extender pelo

utilizador (através de uso de vários procolos bem definidos); Desenvolvimento interactivo, com base em texto com editor integrado e interfaces com menus, com editores e janelas: *MacOS, Windows 95/98/NT, X Window*; Verificação e validação de inconsistencias e erros (análise semântica das regras e dos valores das slots, e dos argumentos das funções); Boa portabilidade entre sistemas; Boa documentação.

Requisitos de Sistema e licenciamento:

- O CLIPS permite operar com vários sistemas: *Windows 95/98/NT*, *MacOS X*, *Unix*. As cópias do CLIPS, os executáveis, a documentação e código fonte podem ser acedidos através do projecto disponível em *SourceForge*, e são sujeitas ao acordo de licensiamento.

Observações Gerais:

O CLIPS é um ambiente para programação de sistemas, com base no conhecimento humano, como por exemplo, sistemas especializados. Pode ser utilizado num conjunto com linguagem de programação ou utilizado separadamente. O sistema CLIPS foi desenvolvido para facilitar a geração de programas que modelizam o conhecimento ou alguma especialização humana [131]. Existem três maneiras de representar conhecimento no CLIPS: a) Regras; b) *Deffunctions* e funções genéricas; c) Programação orientada a objectos. É possível desenvolver programas apenas com a utilização de regras, objectos ou uma combinação dos dois. Regras e objectos formam um sistema integrado, já que regras podem ser utilizadas para o casamento de padrões em factos e objectos.

O CLIPS é uma ferramenta para sistemas especializados, pois trata-se de um ambiente completo para desenvolvimento de sistemas especializados, que inclui um editor integrado e uma ferramenta de apuramento. O terminal de texto (*word shell*) é reservado para a parte do CLIPS que realiza inferência. O modo de execução do *shell* no CLIPS, tanto pode ser por inferência como por *reasoning*. Possui os elementos básicos de um sistema especializado deve ter: a) Lista de factos e instâncias: memória global para os dados; b) Base de conhecimento: que contém todas as regras; c) Motor de inferência: controla a execução das regras. Um programa escrito no CLIPS consiste de regras, factos e objectos.

O motor de inferência, decide quais regras devem ser executadas e quando. Um sistema especializado com base em regras e escrito no CLIPS, é um programa *data-driven* onde os

factos, e os objectos, são eles próprios os dados que estimulam a execução através do motor de inferência. Este é um exemplo de como o CLIPS difere de outros programas convencionais como *PASCAL* e *FORTRAN*. Nestas outras linguagens, a execução pode ser feita sem a necessidade de dados, isto é, as instruções são suficientes para a execução. Já no CLIPS, é preciso dos dados para a execução das regras [132]. O CLIPS tem um desenho que permite uma total integração com outras linguagens tais como C e pode ser usado como uma ferramenta *stand-alone*. Mais ainda, o CLIPS pode ser chamado por outra linguagem, fazer a função que lhe compete e depois fazer o retorno do controlo ao programa que o chamou. O CLIPS tem um bom interface com o utilizador.

Fazer protótipos de regras no CLIPS é útil, porque permite um rápido *feedback*. Mas também, o desenvolvimento de regras e suas alterações podem ser realizados facilmente nesta plataforma. Foi por este conjunto de factores que os autores em [132] escolheram aplicar o CLIPS em primeiro lugar nos seus sistemas.

3.16. IMPACT

O IMPACT é um protótipo desenvolvido para gestão de redes com alarmes em tempo real e análise de falhas [147]. Foi criado em 1991 por *Gabriel Jakobson* e *Mark Weissman* nos laboratórios *GTE Laboratorie*s. Foi desenvolvido em linguagem C, com a utilização de um sistema *X Windows* e com as ferramentas de código fonte aberto, CLIPS e *TclTk*. O IMPACT é um sistema de *shell* especializado em tarefas de correlação de alarmes de rede em tempo real. Foi usado para construir e sustentar várias aplicações de correlação de eventos em redes de telecomunicações com fios e sem fios.

Características e Funcionalidades Especificas:

- Suporta Correlação; Ferramenta sem código fonte aberto (protótipo para uso fechado); Sistema de regras; Faz agregação de eventos; Faz generalização de eventos; Faz especialização de eventos; Correlação causal; Correlação temporal; Executa acções; Permite correlação de eventos em tempo real; Permite propagação de eventos em tempo real; Modelo de execução por "*model-based reasoning*"; Dedica-se as tarefas de manutênção da rede; Orientado para interface gráfica com o utilizador (orientado ao utilizador); Não tem boa documentação.

Requisitos de Sistema e licenciamento:

- O IMPACT foi instalado em sistemas Unix.

Observações Gerais:

O ambiente de desenvolvimento da aplicação IMPACT, contem editores gráficos que descrevem as classes dos elementos de rede, os modelos de configuração de rede, as correlações e as regras de correlação. O IMPACT é um sistema fortemente orientado aos gráficos, com uma componente para visualizar a conectividade de rede, apresentar eventos de rede e visualizar mapas geográficos. Corre em sistemas com plataformas Unix (*IBM, SUN, SGI Indy*). Por exemplo no sistema *SGI Indy R44OOSC*, a performance do IMPACT chega até ao *parsing* e correlação de entre 15 a 30 eventos por segundo [148]. A maioria das fontes de alarme das redes, são as falhas físicas que ocorrem nos elementos de rede *Network Element* (NE). Estas falhas podem ter uma relação de causalidade ou não (isto é, podem ser independentes).

O IMPACT tem um modelo avançado de correlação em tempo real, que tem por base os princípios do modelo "*model-based reasoning*". Foi concebido para analisar hierarquicamente redes e subredes (rede local, rede regional e rede nacional). Um dos passos e tarefas mais importantes no processamento de informação utilizados neste projecto, foi tentar generalizar a informação que é passada dos níveis baixos para os níveis altos e a especialização da informação no sentido contrário, dos níveis mais altos para os mais baixos. Em ambas as duas tarefas de processamento de informação, podem ser feitos através de mecanismos de correlação de eventos. Por exemplo, durante a generalização, um conjunto padrão de eventos pode ser substituído por correlação ou um evento pode ser substituído pela sua superclasse. O sistema de correlação do IMPACT (Event Correlation Engine) foi desenvolvido através da utilização da ferramenta CLIPS.

3.17. SWATCH

O Swatch (Simple watchdog) é um programa que pode ajudar na análise de ficheiros de *log*, fornecendo notificação imediata se as entradas de *log* correspondem a derterminada expressão regular, ou para rever ficheiros de *log* à procura de dados desconhecidos. Foi desenvolvido em linguagem Perl, por Stephen Hansen e Todd Atkins em 1993 [79]. O Swatch é um dos programas de pesquisa mais eficientes e tradicionais, tem muitos adeptos

que o utilizam como programa de pesquisa e análise de logs. O Swatch utiliza um arquivo de ficheiros onde deverão constar as regras de leitura e de classificação dos ficheiros de *log*. Recorrendo ao uso de expressões regulares em Perl, compatíveis com as do padrão *grep*, o Swatch lê o arquivo de ficheiros de *log*, utilizando essas regras, classificando e executando acções premeditadas no arquivo "*${HOME}/.swatchrc"* do utilizador que o executar [153], [154], [155].

Características e Funcionalidades Especificas:

- Suporta Correlação; Ferramenta *Open Source*; Método de pesquisa *Pattern Matching* (com expressões regulares); Utiliza um sistema de regras; Faz filtragem de eventos; Faz supressão de eventos; Faz compressão de eventos; Faz contagem de eventos; Correlação causal; Correlação temporal; Executa acções; Modelo de execução por "*rule-based reasoning*"; Inspecção de diagnóstico e rastreio de problemas; Envia notificações de mail e notificações para a consola; A entrada são os ficheiros de *log* dos sistemas Unix (*syslog*); Capacidade de alerta em tempo real; Boa Documentação; Boa portabilidade.

Requisitos de Sistema e licenciamento:

- Os requisitos para correr o Swatch, é possuir determinadas dependências instaladas no sistema operativo Unix, tais como: os módulos Cpan do Perl. O licensiamento do código fonte do *Swatch* é licenciado segundo os termos do *General Public License* (GNU) e publicado pela *Free Software Foundation*. Que dá permissões de copiar, distribuir e/ou modificar o *Swatch* dentro de certas condições [153].

Observações Gerais:

Por causa da importância dos eventos de *log* como a base de informação do "estado de saúde" dos sistemas Unix. Várias ferramentas foram concebidas para fazer monitorização de eventos de *log*. O Swatch foi a primeira ferramenta O*pen Source*, a fazer monitorização de ficheiros de log. É uma ferramenta à base de regras e bastante popular. O Swatch monitoriza os dados dos ficheiros de *log*, através da leitura de cada mensagem de evento através do *append* em cada linha para o ficheiro de *log* [152]. Compara essas linhas com as regras, onde a parte condicional de cada regra é uma expressão regular (as regras estão depositadas num ficheiro de configuração em formato de texto). Se a expressão regular de

uma certa regra, corresponder à mensagem do evento em determinada linha, o Swatch executa a parte da acção da regra [153]. As acções includem envio de mails, executar programas externos, escrever notificações para o sistema de consola.

O Swatch tem opções para ignorar eventos repetidos, com determinado *treshold* num dado intervalo de tempo. O Swatch foi concebido para monitorizar um grande número de eventos em redes com grande quantidade de servidores e plataformas de trabalho [79]. Mesmo quando os sistemas Unix são capazes de fazer *logging* de muita variedade de informação acerca do estado do sistema. Muitas das vezes, são os locais mais escondidos do sistema Unix que são atacados por intrusões bem sucedidas, por estes, não serem monitorizados regularmente e por serem passíveis de ser apagados ou modificados.

O Swatch foi criado por parte dos autores com os seguintes objectivos [79]: a) configuração de um programa que demorasse pouco tempo a ensinar um administrador de sistema; b) ter um conjunto simples de regras que podem ser imediatamente executadas após receberem, certo tipo de informação; c) permitir aos seus utilizadores definirem as suas próprias acções e permitir que partes das mensagens de entrada sirvam, como argumentos para a acção; d) com o Swatch a correr deve ser possível reconfigurar, ou depois de certo intervalo sem ter que parar, ou fazer *restart* manualmente.

3.18. LOGSURFER

O LogSurfer é um programa de monitorização de eventos em tempo real que verifica contínuamente o sistema de ficheiros de *log*. Reporta a ocorrência de eventos, simplifica a manutenção, auxilia a identificação dos problemas e ajuda à sua resolução [158]. É escrito em linguagem C, o que faz com que seja extremamente eficiente, que é, um factor importante quando existe uma grande quantidade de tráfego de *logs*. O Logsurfer examina as mensagens no ficheiro de *logs*, na forma como estas se inter-relacionam umas com as outras. O Logsurfer é também capaz de modificar as suas regras durante o tempo de execução. Estas propriedades fazem com que Logsurfer detecte padrões complexos nos ficheiros de *log* e actue com base no que encontra.

Todos os administradores de sistemas com grandes clusters, requerem este tipo de análise ao ficheiro de *logs*. O logsurfer põe ao dispor dos administradores de forma muito simples a informação mais útil na análise de um ou mais problemas. Actuando directamente na

informação em vez dos próprios administradores e desta forma, libertando-os para tarefas mais importantes [157].

Características e Funcionalidades Especificas:

- Suporta Correlação; Ferramenta Open Source; Método de pesquisa Pattern Matching (com expressões regulares) e excepções com a segunda expressão regular; Utiliza um sistema de regras; Faz filtragem de eventos; Faz supressão de eventos; Faz compressão de eventos; Faz contagem de eventos; Usa contextos (colecção de mensagens) em vez de linha única; Correlação causal; Correlação temporal; Executa acções; Modelo de execução por "*rule-based reasoning*"; Inspecção de diagnóstico e rastreio de problemas; Envia notificações de mail e notificações para a consola; A entrada são os ficheiros de log dos sistemas Unix (syslog); Capacidade de alerta em tempo real; Flexível e de fácil utilização; Faz o "*shifting*" de ficheiro de logs; Regras dinâmicas podem mudar as acções associadas as mensagem recebidas; Múltiplas reacções numa simples linha com correspondência; Boa Documentação; Boa portabilidade.

Requisitos de Sistema e licenciamento:

- O LogSurfer corre em todas as plataformas [159]: *BSD* (*FreeBSD, NetBSD; OpenBSD, Apple Mac OS X*); *POSIX* (*Linux/BSD/UNIX-like OSes*); *HP-UX, IBM AIX; Linux; SGI IRIX; Solaris.* O licensiamento do código fonte do *LogSurfer* é de acordo com os termos do *General Public License* (GNU), publicado pela *Free Software Foundation*. Que dá permissões de copiar, distribuir e/ou modificar o *LogSurfer* dentro de certas condições [157][159].

Observações Gerais:

Esta aplicação foi concebida para monitorizar qualquer tipo de ficheiro de *log* à base de texto num sistema em tempo real. Uma das características mais importantes do logsurfer é a capacidade de criar novas regras conforme as necessidades [157]. Esta funcionalidade é especialmente útil quando uma mensagem específica, é recorrente, e apenas precisa de ser enviada após um determinado período de tempo. A solução será através de uma nova regra, que ignora essa mensagem por uma determinada janela temporal. Com um sistema de regras dinâmicas pode-se redefinir durante o tempo de execução do programa, quaisquer eventos de interesse.

O LogSurfer é capaz de agrupar vários conjuntos de *logs* de entrada. Por exemplo quando um sistema se reinicializa, normalmente cria um número muito elevado de mensagens de *log*. Neste caso, o LogSurfer pode ser configurado para fazer o agrupamento de todas essas mensagens e reencaminhamento via mail aos administradores do sistema. Estas mensagens de mail têm já toda a informação correlacionada e mais simplificada, advertindo apenas, que o sistema se reinicializou.

O Logsurfer é similar ao programa Swatch no qual assenta, mas oferece um leque de características mais avançadas que o Swatch já não permite [158]. Outra característica importante, é que o e o Logsurfer lida bem com grandes quantidades de eventos diários (mais de 500K eventos por dia).

3.19. SEC

O *Simple Event Correlator* (SEC) ficou disponível no ano de 2001 e foi desenvolvido por Risto Vaarandi. É uma plataforma independente para correlação de eventos [73,74]. Esta plataforma é independente do sistema operativo em que opera e tem por base um sistema de regras (*rule-based reasoning*). A primeira versão do SEC foi aplicada para gestão de falhas de rede [73], mas depois o SEC evoluiu para uma ferramenta de correlação de eventos que é usada em vários domínios de aplicação, tais como detecção de intrusão (IDS), monitorização de *logfiles*, detecção de fraude, etc.

O primeiro objectivo do SEC foi preencher o vazio entre o crescimento interino de soluções caseiras e as soluções comerciais, criando algo leve e fácil de usar. Que pode ser utilizado numa grande variedade de tarefas de correlação de eventos, quer em modo *standalone* ou distribuído, ou seja, integrado com outras aplicações.

Características e Funcionalidades Especificas:

- Suporta Correlação; Ferramenta *Open Source*; Método de pesquisa *Pattern Matching* (com expressões regulares); Utiliza um sistema de regras; Faz filtragem de eventos; Faz supressão de eventos; Faz compressão de eventos; Faz contagem de eventos; Tem o conceito de contextos; Correlação causal; Correlação temporal; Executa acções; Modelo de execução por "*rule-based reasoning*"; Inspecção de diagnóstico e rastreio de problemas; Envia notificações de mail e notificações para a consola; A entrada são

os ficheiros de *log* dos sistemas Unix (*syslog*) ou qualquer outro tipo; Capacidade de alerta em tempo real; Boa Documentação; Boa portabilidade.

Requisitos de Sistema e licenciamento:

- Os requisitos para correr o SEC, é possuir apenas algumas dependências instaladas no sistema operativo Unix, tais como: o módulo Perl. No caso das últimas versões do SO Ubuntu, este já vem incluído no pacote de instalação. O licensiamento e distribuição do código fonte do *SEC* é segundo os termos do GNU da *General Public License* (GNU) [53].

Observações Gerais:

O SEC é uma ferramenta de correlação de eventos desenvolvida com o propósito de ser independente da plataforma em que opera. O Perl corre praticamente em todos os sistemas e tornou-se uma norma integrante de muitos sistemas operativos [91]. As aplicações escritas em Perl trabalham quase tão rápido como os programas escritos em C. O SEC não precisa de muito espaço em disco, o seu tamanho ronda os 160KB e a sua configuração é gravada em ficheiros de texto que apenas necessitam de mais uns Kilobytes. Como é todo escrito em Perl, pode ser usado instantâneamente logo após a sua distribuição ser desempacotada, não precisa de preparações adicionais, tais como compilação e ligar às fontes (*linking the source*).

O SEC recebe os seus eventos externos através de um ficheiro de fluxo (*file stream*) e ao executar comandos de *shell* especificamente configurados pelo utilizador, produz efeitos à sua saída (*Standard ouput*). Os ficheiros regulares, chamados *pipes* servem como entradas normalizadas (*standard input*). Estes tipos de canais de fluxo de informação são correntemente suportados como entradas no SEC. Desta forma, qualquer tipo de aplicação que esteja apto para escrever o seu conteúdo numa *file stream* pode ser integrado juntamente com o SEC. As aplicações que têm a gestão de eventos através de uma *Application Programming Interface* (API), podem também ser integradas através de simples *plugins* que transmitem chamadas da API para ler a *stream* de eventos da aplicação e copiá-la para um ficheiro ou para a sua saída normalizada (*standard ouput*), o SEC tem no seu pacote de instalação (*package*) um *pugin* para o *HP OpenView ITO*.

Independentemente do formato utilizado, o SEC utiliza expressões regulares para reconhecer e manipular qualquer tipo de eventos que aparecem à sua entrada. Esta forma facilita a configuração do SEC, já que muitas das ferramentas do sistema operativo Unix, como o *grep, o sed*, *o find* e o *awk*, confiam nas expressões regulares para fazer as suas tarefas. Muitos dos administradores de sistemas já estão familiarizados com a linguagem de expressões regulares. A configuração do SEC pode ser gravada em ficheiros de texto que podem ser criados e modificados com qualquer tipo de editor de texto. Cada ficheiro de configuração contém uma ou mais regras e outros conjuntos de regras de outros ficheiros podem ser aplicados lógicamente em paralelo.

O SEC além de permitir a condição de correspondência (*matching*) do evento, permite fazer a respectiva acção a esse acontecimento, ou seja, a regras definem uma lista específica de acções que podem ser desencadeadas após a correspondência, opcionalmente o SEC tem a capacidade de gerar expressões booleanas chamadas de "contexto". Os contextos do SEC, representam a confirmação que o SEC aprendeu durante todo o processo de correlação de eventos. Cada contexto tem o seu próprio tempo de vida (finito ou infinito) e permitem activar ou desactivar regras dinâmicamente durante o tempo de execução. Os contextos são uma das características mais poderosas do SEC. A noção de contexto pode ser de difícil reconhecimento mental, já que, eles são instâncias lógicas de dados e agem, quase como variáveis Perl, brotando à existência à medida que são criados. A sua principal característica é a forma como interagem com as regras durante a sua existência lógica.

É possível desenvolver cenários bastante complexos e confusos de correlação de eventos ao combinar a correspondência da amostra padrão (*pattern matching)*, com contextos e criação e manipulação de regras [93]. O SEC é conceptualmente concebido para utilizar regras "*model base reasoning*".

3.20. ESTUDO E CONCLUSÕES SOBRE AS FERRAMENTAS ABORDADAS

Depois do estudo de nove ferramentas de correlação de eventos: *Snort, SnortSnarf, SnortCenter, HP OPenView ECS, CLIPS, IMPACT, Swatch, LogSurfer e SEC*. Foi elaborado uma tabela (tabela 3), que permite visualizar as características específicas de cada uma ferramenta e também fazer uma análise das principais diferenças.

O objectivo deste trabalho é análise dos eventos de rede geridos pelo NMSIS, tal como o estado dos elementos de rede e dos serviços. Foram ponderadas várias formas de como obter a informação dos eventos do NMSIS, após o estudo do NMSIS, chegou-se à conclusão que não é possível fazer a análise dos eventos internos do NMSIS (do tipo eventos de *log* do sistema), mas sim a informação do estado dos dispositivos de rede.

Para atingir, esse fim, uma das formas possiveis foi criar um script de raíz (*daemon*) que pudesse fazer querys as tabelas do NMSIS e pudesse ele próprio ter a condição de criar um ficheiro do tipo *log* contendo toda a informação do estado dos objectos (elementos de rede e serviços) que se pretende monitorizar. Estas tarefas executadas pelo novo script a correrr como um *daemon* vai ser exemplificado mais a frente. Com este *daemon* é possível colectar toda a informação de interesse do NMSIS e disponibilizar os eventos à entrada do novo sistema SCEN.

O novo motor de correlação terá como função analisar os eventos gerados pelo *daemon* e remetido à sua entrada.

Tabela 3 Características e funcionalidades genericas de uma ferramenta de correlação de eventos.

ID	Caracteristicas Funcionais	Snort	SnortSnarf	SnortCenter	HP OPenView ECS	CLIPS	IMPACT	Swatch	LogSurfer	SEC
1	Suporta Correlação				•	•	•	•	•	•
2	Suporta Correlação limitada		•	•						
3	Ferramenta Open Source	•	•	•		•		•	•	•
4	Ferramenta com código de uso fechado ao publico						•			
5	Ferramenta comercial				•					
6	Método de pesquisa Pattern Matching	•						•	•	•
7	Modelo de execução por rule-based reasoning				•	•		•	•	•
8	Modelo de execução por model-based reasoning						•			
9	Modelo de execução por inferência					•				•
10	Utiliza um sistema de regras	•	•	•	•	•	•	•	•	•
11	Cria regras pessoais e modifica as regras existentes			•						
12	Regras dinâmicas podem mudar as acções associadas as mensagem recebidas								•	•
13	Permite testar as regras definidas em simulate mode				•					
14	Suporta Templates de regras para configuração de múltiplos sensores			•						
15	Método de pesquisa Pattern Matching (com expressões regulares)							•	•	•
16	Método de pesquisa Pattern Matching e excepções com a segunda expressão regular									
17	Faz filtragem de eventos		•	•	•			•	•	•
18	Faz supressão de eventos				•			•	•	•
19	Faz compressão de eventos							•	•	•
20	Faz contagem de eventos							•	•	•
21	Faz agregação de eventos						•			
22	Faz generalização de eventos						•			
23	Faz especialização de eventos						•			
24	Tem o conceito de contextos								•	•
25	Correlação causal						•	•	•	•
26	Correlação temporal						•	•	•	•
27	Permite correlação de eventos em tempo real				•			•	•	•
28	Permite propagação de eventos em tempo real						•			
29	Definem-se “circuitos de correlação” para o processamento de eventos				•					
30	Correlações simples através da interligação das primitivas de um nó				•					
31	Tecnicas de feedback dos dados do circuito de correlação				•					
32	Executa acções (conceito de acção e de reacção)						•	•	•	•
33	Múltiplas acções numa simples linha com correspondência								•	•
34	Envia notificações de mail	•						•	•	•
35	Envia notificações de alerta, por socket Unix ou por mensagens *Winpopup* Windows	•								
36	Análise de Protocolos	•								
37	A entrada são os ficheiros de log dos sistemas Unix (syslog)		•					•	•	
38	Entrada são os ficheiros de alerta do Snort no directório (\log) ou a base de dados do Snort		•							

39	A entrada pode ser qualquer tipo de ficheiro ou aplicação									•
40	Capacidade de alerta em tempo real	•			•	•		•	•	•
41	Utiliza armazenamento de eventos em ficheiros (logs)	•			•					
42	Faz "shifting" de ficheiros de logs								•	
43	Dedica-se as tarefas de manutênção da rede				•		•	•	•	•
44	Inspecção de diagnóstico e rastreio de problemas	•	•				•	•	•	•
45	Geração de programas que modelizam o conhecimento ou alguma especialidade humana				•					
46	Plataforma de desenvolvimento e produção para sistema de regras e/ou objectos					•				
47	Facil de integrar com outras linguagens (C, Java, FORTRAN and ADA)					•				
48	Facil de extender pelo utilizador (através de uso de vários procolos bem definidos)					•				
49	Verificação e validação de inconsistencias e erros					•				
50	Sensor Agente - Múltiplos daemons em múltiplas interfaces de rede			•						
51	Consola de gestão e Sensor Agents para Linux, BSD, *NIX, Windows			•						
52	Encripatção SSL entre o sistema de gestão e os sensores agentes			•						
53	Desenvolvimento Interactivo, com editores e janelas: MacOS, Windows, X Window					•				
54	Orientado para interface gráfica (em java), orientado ao utilizador						•			
55	Mecanismos de visualização e de análise dos dados;	•								
56	Saída em Web (HTML)		•	•						
57	Boa portabilidade entre sistemas	•	•	•		•		•	•	•
58	Boa Documentação	•		•	•	•		•	•	•

Como se pode constatar, o número de características neste tipo de ferramentas é bastante elevado, muitas destas características são importantes para o trabalho que se pretende fazer, outras são menos importantes, umas são mais específicas e outras mais genéricas. Destas nove ferramentas que permitem a correlação de eventos: *Snort*, *SnortSnarf*, *SnortCenter*, *HP OPenView ECS*, *CLIPS*, *IMPACT*, *Swatch*, *LogSurfer e SEC*, será feito o levantamento e escolha das suas características mais importantes e que pretendemos ver implementadas no sistema SCEN.

O objectivo do trabalho na escolha destas ferramentas é que estas permitam analisar o conteudo dos eventos gerados através de um *daemon* que tem como função principal, colectar para um *pipe* ou para um ficheiro de *log* toda a informação do estado dos objectos (elementos de rede, etc) recolhida do NMSIS, a cada instante.

A tabela 4, tem já essa abordagem para termos um estudo mais direccionado ao que se pretende ver implementado na nova ferramenta.

Tabela 4 Características e funcionalidades mais importantes de uma ferramenta de correlação

ID	Caracteristicas Funcionais	Snort	SnortSnarf	SnortCenter	HP OPenView ECS	CLIPS	IMPACT	Swatch	LogSurfer	SEC
1	Suporta Correlação				•	•	•	•	•	•
3	Ferramenta Open Source	•	•	•		•		•	•	•
6	Método de pesquisa Pattern Matching	•						•	•	•
7	Modelo de execução por rule-based reasoning				•	•		•	•	•
9	Modelo de execução por inferência					•				•
10	Utiliza um sistema de regras	•	•	•	•	•	•	•	•	•
12	Regras dinâmicas podem mudar as acções associadas as mensagem recebidas								•	•
15	Método de pesquisa Pattern Matching (com expressões regulares)							•	•	•
17	Faz filtragem de eventos		•	•	•			•	•	•
18	Faz supressão de eventos				•			•	•	•
19	Faz compressão de eventos							•	•	•
20	Faz contagem de eventos							•	•	•
21	Faz agregação de eventos						•			•
22	Faz generalização de eventos						•			•
23	Faz especialização de eventos						•			•
24	Tem o conceito de contextos								•	•
25	Correlação causal						•	•	•	•
26	Correlação temporal						•	•	•	•
27	Permite correlação de eventos em tempo real				•			•	•	•
32	Executa acções (conceito de acção e de reacção)						•	•	•	•
33	Múltiplas acções numa simples linha com correspondência								•	•
34	Envia notificações de mail	•						•	•	•
39	A entrada pode ser qualquer tipo de ficheiro ou aplicação									•
40	Capacidade de alerta em tempo real	•			•	•		•	•	•
41	Utiliza armazenamento de eventos em ficheiros (logs)	•			•					
42	Faz "shifting" de ficheiros de logs								•	
43	Dedica-se as tarefas de manutênção da rede				•		•	•	•	•
44	Inspecção de diagnóstico e rastreio de problemas	•	•				•	•	•	•
56	Saída em Web (HTML)		•	•						
57	Boa portabilidade entre sistemas	•	•	•		•		•	•	•
58	Boa Documentação	•		•	•	•		•	•	•
Número de Caracteristicas: 31		**9**	**6**	**6**	**10**	**8**	**10**	**20**	**24**	**28**

Após analise da tabela 4, podemos constatar que as ferramentas comerciais de correlação de eventos, como a ferramenta *HP ECS* [82] que estudamos, ou outros sistemas como o *SMARTS* [83], o *NerveCenter* [84]. Todos têm muito sucesso e são usados no mundo inteiro por inúmeras empresas, mas sofreram vários recuos ao longo do tempo, por

diversas razões. Uma das razões é que estes sistemas são normalmente soluções pesadas que têm um design complicado bem como a interface com o utilizador. A sua complexidade e tamanho fazem destas ferramentas quase impossíveis de trabalhar em pequenas redes e em pequenas tarefas de correlação de eventos, especialmente em nós de rede com capacidade de hardware limitada (ex: análise e correlação de *logfiles* numa plataforma com pouco disco, memória e fraco processamento do CPU). Como estes sistemas são na sua maioria comerciais, acabam por ser *platform-dependent* e são uma boa razão para sua exclusão no âmbito deste trabalho.

O CLIPS [89], que é um sistema para especialistas num ambiente de criação à base de regras (*rule-based expert systems*), tem muito sucesso quando usado para construir sistemas com correlação de eventos [75][90]. Mas, como não é uma ferramenta específica para correlação de eventos o CLIPS, não é o tipo de ferramenta que se pretende ver integrado e implementado no sistema que se pretende criar no âmbito deste trabalho.

As funções de gestão de segurança e detecção de intrusões [76], [77], podem ser da responsabilidade de ferramentas como o Snort juntamente integrado com o SnortSnarf ou com o SnortCenter [78], que aplicam contagem de eventos para detectar *portscans*, (*hakers* a fazer *snifering* das portas abertas), etc. Entre outras caracteristicas, têm técnicas de correlação de eventos tais como administração de sistemas com análise de registo de eventos (*logfile analysis*). Embora sejam ferramentas muito úteis em sistemas IDS, não são o tipo de ferramentas que se pretende ver implementada com o NMSIS, já que o NMSIS é uma ferramenta de gestão de redes com protocolo SNMP e não um *Snifer* de protocos que correm na rede. Por esse motivo estas ferrametas foram excluídas do projecto.

Algumas tarefas de análise de *logfile* podem ser executados com a utilização de ferramentas como o *Swatch* [79], o *Logsurfer* [80] ou o SEC. O *Swatch* e o *LogSurfer* são ferramentas com os mesmos princípios funcionais. O *Swatch* é escrito em linguagem Perl e o *LogSurfer* em linguagem C. O Logsurfer lida bem com grandes quantidades de eventos por dia, tal como o SEC, enquanto, que o Swatch, já tem mais dificuldades a lidar com grandes quantidades de tráfego de eventos. O *Swatch* e o *Logsurfer* suportam algumas operações de correlação de eventos, mas com capacidades de correlação ainda um pouco limitadas.

Um artigo [81] aponta para as fragilidades das ferramentas de monitorização de *logfiles*, e discute a importância de estender as técnicas de monitorização de *logfile* a um modelo de aproximação mais complexo e heuristico, incluindo a correlação de eventos. Mesmo comungando algumas das caracteristicas de base das ferramentas *Swatch* e *LogSurfer*, a ferramenta SEC [74], não está presa apenas análise de ficheiros de *logs*.

O seu conceito assenta na característica de ser *plataform independent* e na capacidade de trabalhar com qualquer tipo de ficheiro de entrada, basta que para isso, o programa exterior possa escrever os seus dados num *standard input* ou através de uma API. O SEC não é uma ferramenta IDS tal como o *Snort, SnortSnart, SnortCenter*, utilizada exclusivamente para segurança. É uma ferramenta mais genérica com capacidades de inferir um nível bastante elevado na complexidade das relações entre os eventos de entrada (correlação). É uma ferramenta aberta ao exterior e como tal não tem qualquer tipo de encargos. É de todas as outras ferramentas a mais completa e genérica no âmbito das suas capacidades.

Não foi escolhida exclusivamenta pela quantidade de características que se pretendiam ver implementadas no novo sistema de correlação, foi escolhida porque de todas as caraceristicas disponibilizadas pelas ferramentas, deu-se especial importância às ferramentas:

- Com a capacidade de análise, verificação e rastreio de informação em *file streams* ou ficheiros.

- As que tenham um motor de inferência capaz de realizar a verificação de informação, de poder processar relações causais e temporais entre eventos e reagir quando esses eventos acontecem.

- As que possuam a capacidade de suportar todas a funcionalidades de filtragem e supressão de informação repetitiva, a capacidade de fazer generalização e especialização dos eventos.

- As que Incorporem um sistema de regras que trabalhe essencialmente com condições e respectiva resposta em função do valor de entrada, este tipo de aproximação ou técnica de correlação de eventos torna as soluções menos complexas do ponto de vista funcional.

- As que tenham a capacidade de ter até vários tipos de acções, e várias acções a decorrer simultâneamente no mesmo instante de execução, para apenas uma condição validada, o que torna o sistema com grande capacidade de resposta e de grande flexibilidade face aos vários tipos de entradas possíveis. É mais uma vantagem para o sistema.

- As que puderem dar uma resposta adequada ao tipo de falhas, através de notificações em tempo real (email, etc) aos administradores de gestão de rede.

- As que usem a capacidade de ter um sistema de regras que permite lançar eventos internos que possam ser processados por outras regras.

- E por fim, as que podem usufruir de eventos chamados contextos com determinado limite de tempo de execução e fazem *triggering* a outras regras que inicializam instantâneamente.

Pela sua eficássia real e grande flexibilidade chegou-se à conclusão que é a ferramenta mais funcional e de eleição para o sistema que se quer ver implementado. Neste contexto será feito no próximo capítulo uma descrição mais aprofundada do modelo de funcionamento da ferramenta SEC.

3.21. SEC - MODELO FUNCIONAL

3.21.1. MODELO E OPERAÇÕES DISPONÍVEIS A UTILIZAR NO SEC

O modelo do SEC tem como base os sistemas do tipo *Rule-based reasoning* com um amplo conjunto de regras que fazem a verificação através de um motor de inferência. Utiliza a chamada memória de execução que serve para salvar informação durante o período de execução dos processos de verificação. Quando se pretende criar determinadas regras no SEC, algumas questões podem ser levantadas, do tipo "quais as regras que queremos utilizar e implementar". Na altura de implementação de um novo processo no SEC, podemos escolher vários tipos de regras que estão actualmente disponíveis em termos de configuração.

O SEC suporta várias formas de compressão, supressão, filtragem, contagem, relação causal e temporal e operações de *clustering*. Ao combinar várias regras, podemos obter inúmeras variantes de todas as operações de correlação de eventos [123]. O ficheiro de configuração do SEC consiste numa serie de definições de regras, as definições são feitas

por linha, com o espaço em branco e espaço de "TAB" usados como separadores de campo. Todas as definições das regras têm as seguintes partes:

- **Tipo de regra (Rule Type)** – *Single, SingleWithScript, SingleWithSuppress, Pair, PairWithWindow, SingleWithThreshold, SingleWith2Thresholds, Suppress,* e *Calendar*.

- **Comportamento depois da Correspondência** – Especifica se a procura de correspondência (*matching*) deve continuar ou não, depois de encontrado a correspondência entre a regra corrente e a linha de entrada. Uma das *strings TakeNext* e *DontCont* deve ser especificada como valor.

- **Tipo e padrão de amostragem (pattern)** – Especifica o padrão de amostragem (*pattern* =) que a linha de entrada vai ser comparada para descobrir o evento. As expressões regulares (tipo *RegExp*) e strings (*tipo SubStr*) podem ser usadas como padrão de amostragem (*pattern* =). Se o tipo de padrão é seguido de um número, o número especifica quantas linhas de entrada serão usadas para a comparação.

- **Descrição de Evento** – Texto descritivo do evento descoberto.

- **Acção** – Define a acção que vai ser executada quando o evento é descoberto. As acções incluem execução de comandos de shell, criação de contextos, apagar de contextos, e fazer *reset* a operações de correlação (existem regras que fazem *reset* de contagem).

- **Contagem e Restrições Temporais** – Restrições opcionais para implementação de operações de correlação, como contagem e relação temporal.

- **Contextos** – A opção contexto existe quando uma regra é considerada válida durante o tempo de execução.

3.21.2. TIPOS DE REGRAS SUPORTADAS PELO SEC

O SEC foi concebido para sustentar vários tipos de regras [74]:

- **Single** – Faz correspondência ao evento de entrada e executa uma lista de acções.

- **SingleWithScript** – Faz correspondência ao evento de entrada e executa uma lista de acções, se um *script* ou programa externo (ex: uma *query* até uma base de dados) e retorna um certo valor de saída (ex: 0,1). O *script* ou programa externo poderá ser provido com o nome dos contextos na sua entrada.

- **SingleWithSuppress** – Faz correspondência ao evento de entrada e executa uma lista de acções, mas ignora novos eventos com correspondência durante uma janela temporal.

- **Pair** – Faz correspondência ao evento de entrada e executa uma lista de acções imediatamente e durante os próximos τ segundos, ignora os próximos eventos com correspondência até chegar qualquer outro evento de entrada. À chegada do segundo evento com correspondência, executa outra lista de acções.

- **PairWithWindow** – Faz correspondência ao evento de entrada e espera durante os próximos, τ segundos, até chegar qualquer outro evento de entrada. Se esse evento não aparece durante o tempo dado na janela, executa uma lista de acções. Se o evento aparece durante o tempo dado na janela, executa outra lista de acções

- **SingleWithThreshold** – Conta as correspondências de um mesmo evento de entrada durante uma janela de τ segundos e se um determinado *threshold*, "n" fôr excedido, executa uma lista de acções. A janela é tipo *Sliding Window.*

- **SingleWith2Thresholds** – Conta as correspondências de um mesmo evento de entrada durante uma janela de τ segundos e se um determinado *threshold*, "n" fôr excedido, executa uma lista de acções. A contagem continua após a execução, quando o número de eventos não fôr superior a "n" durante a janela, outra lista de acções é executada. Ambas as janelas de correlação de eventos são do tipo *Sliding Window,* quer dizer que o tempo de inicio de uma janela move-se para o tempo da próxima janela se o tempo τ da primeira correspondência expirar.

- **Suppress** – Faz correspondência ao evento de entrada e suprime-o.
- **Calendar** – Executa uma lista de acções a uma determinada hora especifica.

3.21.3. TIPOS DE ACÇÕES SUPORTADAS PELO SEC

As acções do SEC não foram desenvolvidas apenas para gerar eventos de saída, mas sim para fazer as regras interagir umas com as outras Para guardar e gerir, eventos internos e para ligar sistemas externos, o SEC suporta vários tipos de regras [74], das quais se destacam:

- **none** – Não faz nenhuma acção.
- **logonly** – Faz *log* de uma mensagem.
- **write** – Escreve a linha para um ficheiro ou para um *pipe*.
- **shellcmd** – Executa um programa ou *script* externo.
- **pipe** – Executa um programa ou *script* externo e o resultados é realimentado à sua entrada.
- **spawn** – Executa um programa ou *script* externo e que fornece o SEC com eventos de entrada adicionais (ex: correr "tail –f" num ficheiro e iniciar um *subdeamon* do SEC para correlação de eventos, etc)
- **create** – Cria um contexto e um conjunto de parâmetros opcionais (tais como, tempo de vida).
- **set** – Define os parâmetros de um contexto com novos valores.
- **delete** – Apaga um contexto.
- **add** – Associa um evento com um contexto.
- **report** – Permite que os eventos adicionados aos contextos sejam visíveis no exterior para processamento externo.

- **event** – Gera um novo evento de entrada que se pode fazer correspondência (*be matched*) com outras regras.

- **reset** – Cancela a operação de correlação de eventos

Ao combinar várias regras com listas de acções apropriadas e com contextos, podemos definir, esquemas de correlação maiores e mais complexos.

3.21.1. EXEMPLO DE UM PROCESSO DE CRIAÇÃO DE UMA REGRA

Vamos ver um processo de criação de uma regra, com apenas um pequeno e muito simples exemplo de configuração utilizando o SEC. O SEC usa um arquivo de configuração e utiliza à sua entrada os dados de um ficheiro de texto ou de uma entrada normalizada, como uma *file stream* (mais conhecido como *pipe*, ou ficheiro do tipo *fifo*). Irá ser feito uma demonstração de algumas etapas a seguir sempre que se quer configurar uma regra. Para isso, foi criado um exemplo típico de uma regra.

Começa-se por criar um ficheiro de texto, por exemplo "SCEN.01.conf" e copia-se para lá o seguinte texto.

Exemplo de um ficheiro de configuração do SEC (SCEN.01.conf):

```
# Exemplo SCEN.01.conf
# Reconhecer um padrão e fazer log
#
type=Single             *1
ptype=RegExp            *2
pattern=foo\s+(\S+)     *3
desc=$0                 *4
action=logonly          *5
```

Este exemplo ilustra o seguinte:

1** ***Single é o tipo de regra. O SEC inclui diversos tipos de regras que são úteis na correlação de eventos. Esta regra é do tipo *Single*.

2** ***RegExp (*Regular Expressions*) é um tipo de padrão de correspondência ou *SubStr* para correspondência em simples cadeias de caracteres.

3** ***foo\s+(\S+) é o padrão actual, neste caso um padrão de expressão regular do perl (falaremos mais a frente). Este padrão corresponde à palavra "foo" seguido por um ou mais espaços, seguidos de um ou mais caracteres de não-espaço, tais como *bar, grok,* ou "1:443z--?". Para este tipo de correspondência utilizando expressões regulares foi estudado as expressões regulares tais como as de Nicholas Clarks [96] entre outras [97], [98].

4** ***desc é uma definição de variável para a descrição do padrão de amostragem. Neste caso *$0* que é uma variável numerada do perl, é definida para guardar todo o padrão de amostragem reconhecido ou correspondido.

5** ***action é uma instrução que descreve a acção executada quando o padrão é reconhecido. Neste caso, a ação *logonly* simplesmente grava o padrão de amostragem para o ficheiro de *log* se o ficheiro fôr indicado na linha de comando, senão fôr, grava para o ficheiro de saída escolhido por defeito. Poderiamos ter dado uma acção para escrever em determinado ficheiro a data e hora interna do SEC e escrever uma mensagem (ex: action=write out.txt %t "$0 escrito com sucesso").

Depois de gravado o ficheiro SCEN.01.conf é executado o seguinte commando:

```
sec -conf=SCEN.01.conf -input=-
```

Esta linha de comando muito simples apenas indica qual o ficheiro de configuração do SEC e qual o caminho para o ficheiro de entrada. Neste caso particular a entrada é o *stdtin* (*standard Input*) ou seja, o periférico teclado, poderia ser um ficheiro, bastava par isso colocar o caminho (ex: -input=/home/casilva/teste) e depois fazer *cat* de mensagens de *echo* para este ficheiro, existem inúmeras opções. Com este exemplo o SEC responde da seguinte forma: sempre que escrevemos *foo* seguido por um ou mais espaços, seguidos de um ou mais caracteres de não-espaço, ele faz o logging para o *stdout* e envia para o ecrã. Podemos ver ínumeros exemplos contidos nos artigos da página [53].

Nos capítulos seguintes, existirão exemplos práticos de regras propostas e descritas para os eventos do NMSIS.

3.21.2. PERFORMANCE E APLICAÇÃO DO MODELO SEC

O SEC é uma ferramenta com uma modesta exigência em termos de velocidade de CPU, pode ser utilizado e instalado em plataformas antigas (foi bem sucedido em sistemas Linux com processador Intel 80486 e com 16MB de memória). Mesmo quando existem centenas de operações de correlação de eventos e contextos, que são activados simultâneamente, e guardados na memória activa de trabalho, mesmo assim, o SEC corresponde muito bem. O programa consome menos de 5 MB de memória na maioria das arquitecturas (depende do sistema operativo).

Durante uma experiencia em que o SEC foi instalado num servidor 1GHz PIII. O teste decorreu com duração de 17,5 dias (1523599 segundos), na sua configuração o SEC tinha 57 regras, com uma média de 47,5 eventos de entrada por segundo, 9,2% dos 72390360 eventos de entrada houve total correspondência e foram correlacionados todos os eventos. Durante esta experiência o SEC consumiu 4,7% do tempo de CPU [73], [74].

O autor do SEC recebeu inúmeros pedidos de informação de várias empresas, das quais destacam-se as seguintes, tabela 5.

Tabela 5 Resultados do *survey* de utilizadores do SEC

Type of the company	Location	Description of the managed network	How SEC is applied	Advantages of SEC over other event correlation systems
Banking card authorization center	Europe	30 servers, routers, and firewalls	Event correlation engine for NMS and IDS, logfile monitoring and system monitoring. An important application of SEC is fraud detection.	Straightforward, easy, and transparent configuration and rule definition system.
Technology-based marketing agency	US	600 nodes across US and UK	Gather and correlate service issues from Cisco CSS content switches.	Power and control in the amount you choose.
Financial institution	US	6000 workstations, 400 servers, 350 switches, 250 routers (distributed over US plus 5 other countries)	Used as a central event correlation engine for HP OpenView NNM. Also used for central monitoring of *syslog*-messages from Cisco devices.	More flexible and customizable than other event correlation systems.
Retail sales of consumer electronics	US	8000 managed nodes; the company WAN covers continental US, Alaska, Hawaii, and US territories	Network management with HP OpenView, logfile monitoring, dynamic webpage generation, etc.	SEC provides a low cost and efficient method to plug in event correlation and event management into HP OpenView.
Telecommuni-cations Carrier/ Provider	US	One of the largest international networks in the world	Logfile monitoring (collect and interpret alarms at call centers, and send a notification to a national support team).	Good level of control over monitoring triggers.
Network consulting	Global (more than 30 offices in US, Europe, and Asia)	SEC is used in the US network of a major European car manu-facturer (100 routers, 300 switches)	Used as a correlation engine for Cisco DFM platform and for Snort IDS.	Provides event correlation without significant programming resources, runs on multiple platforms, integrates well with external scripting languages.
Software development, IT consulting and services	Global (offices in Europe, US, Asia, Australia, South-America)	Global network, spread worldwide across the globe	Used as a prototype for event correlation experiments.	Free download status.

O SEC já foi usado em inúmeras plataformas, como Linux, Solaris, HP_UX, AIX, FreeBSD, Tru64 UNIX e Windows. Foi aplicado em diversos domínios como ambientes com inúmeras avarias de rede e de performance, de intrusão e detecção, de monitorização e análise de ficheiros de *log*. Foi também utilizado para investigação académica [92].

3.22. RESUMO DO CAPITULO

Neste capítulo começamos por referir que o conceito de correlação de eventos pode ser definido pelo coeficiente de correlação que indica a força e a direcção da relacção linear entre duas variáveis aleatórias. Foram abordados as permissas da correlação de eventos, ou seja, diagnosticar a raiz do problema que causa falhas, para isso é preciso recorrer a técnicas e métodos de filtragem de eventos (que inundam a rede) correlacionando-os num pequeno número ou um único evento conceptual. Para que garantir a eficássia do sistema, este deverá ser correcto (acertar com alta probabilidade na raiz da causa do problema) e deverá ser ideal (o sistema deve inferir um pequeno número de raízes das causas que podem explicar todos os eventos detectados).

Foram abordadas questões a respeito dos eventos e falhas, os eventos estão associados a um objecto no qual eles ocorrem. O objecto é uma entidade com determinado estado, modelizado como uma colecção de parâmetros cujos valores podem mudar com o tempo. A falha é um evento que está associado a um estado anormal da rede. Eventos podem ser nativos ou compostos, são nativos se forem gerados internamente e são directamente observáveis. Poderão ser compostos se não são directamente observáveis. Algumas das características mais importantes da correlação de eventos e falhas, são a detecção e isolamento de falhas, que é uma das funções mais importantes para descoberta da causa mais provável, que está na raiz do problema. A filtragem e supressão de eventos, é outra característica muito importante que o sistema deve possuir, deve reportar apenas alarmes primários, tais como o estado de switchs ou routers (rı, sı), enquanto suprime os eventos secundários e potêncialmente numerosos de falhas do tipo *"falha na conexão*", etc. Outra característica não menos importante é o ajuste de performance, que é quando se consegue isolar eficientemente a raiz do problema que provoca as falhas.

Posteriormente foi discutido em pormenor o assunto da correlação causal, este tipo de relação, pode-se considerar como a mais simples que pode existir entre os eventos. A correlação causal e filtragem, é a resposta para chegar à raiz do problema. Determinado

evento “a” é causa do evento “a’ ”, ou seja a → a’. Posto este tipo de relação, pode-se desenhar gráficos em forma de um esqueleto ou árvore de causa – efeito. Estes gráficos ilustram de uma forma directa e não cíclica, a representação de eventos através dos nós e a representação das relações de causalidade entre os nós, através das setas. É também conceptualmente mais fácil definir um esquema de correlação e filtragem através deste tipo de gráfico.

Outro tipo de correlação é a temporal, neste caso podem ser estabelecidas correlações entre eventos com base num determinado instante temporal. Os eventos causais são obtidos através da interpretação dos eventos como se de proposições que tem os valores de forma booleana em verdadeiro ou falso (1 ou 0) se tratassem, indicando a sua ocorrência. Para representar uma relação temporal entre eventos é preciso definir a interpretação dos eventos para incluir informação acerca do instante temporal a que cada evento ocorreu. Podemos interpretar um evento como sendo função do tempo τ com os estados {verdadeiro, falso} e com instantes de ocorrência definidos pelos argumentos (instantes temporais) no qual eles são verdadeiros. Muitos autores falam sobre vários tipos de operações que podem ser executadas nos eventos, durante o processo de correlação, estes tipos de operações são: a compressão; supressão; a filtragem; a contagem; o dimensionamento; a generalização; a especialização; a relação temporal; o *clustering*, etc.

Seguidamente foram estudados os métodos e técnicas adoptadas em investigação de inteligência artificial para realizar operações de correlação. Nas técnicas de correlação de eventos é sempre necessário ter algumas premissas que servem de base ao raciocínio e que permitem chegar a determinado tipo de conclusões. Estas aproximações foram separadas em várias categorias distintas. A correlação de eventos pode ser obtida através de vários modelos, ou mesmo através da sua combinação, estes catogorias são: *rule-based reasoning* (RBR); *model-based reasoning* (MBR); *case-based reasoning* (CBR); *code-base (codebook) correlation* (CC); *State Transtion base (STB); Probabilistic Finite State Machine (PFSM); Network Element Dependency Graph (NEDG); Hybrid approach*.

Posto isto, foram pesquisadas e estudadas várias ferramentas que actualmente tem operações de correlação. O sentido desta pesquisa foi orientado na procura de ferramentas com código fonte aberto, segundo os termos do *Free Software Foundation* (FSF) especifico das ferramentas do sistema operacional *General Public License* (GNU), contudo foram também abordadas ferramentas comerciais. As ferramentas de correlação eventos

estudadas, foram nove: *Snort, SnortSnarf, SnortCenter, HP OPenView ECS, CLIPS, IMPACT, Swatch, LogSurfer e SEC* para ser feita a escolha de qual delas será utilizada integrada com o NMSIS (NMSIS). Posto o estudo das nove ferramentas de correlação de eventos Foi elaborado uma tabela geral para visualizar as diferenças das suas características específicas. Constatatou-se que o número de características das ferramentas era muito elevado, por isso restringiu-se o estudo apenas as características pretendidas e fez-se a escolha. Esta escolha recaíu sobre a ferramenta escrita em linguagem Perl, chamada Simple Event Correlator (SEC). Por esse motivo, foi preparado um trabalho descritivo mais pormenorizado das características desta ferramenta.

O SEC utiliza regras que, suportam operações de compressão, supressão, filtragem, contagem, relação temporal e operações de clustering. Podem ser configuradas combinações de várias regras com inúmeras variantes de todas as operações de correlação de eventos. Tem um ficheiro de configuração que consiste numa serie de definições de regras, uma definição por linha, com o espaço em branco e espaço de "TAB" usados como separadores de campo. Todas as definições das regras a seguintes constituição: Tipo de regra (*Single, SingleWithScript, SingleWithSuppress, Pair, PairWithWindow, SingleWithThreshold, SingleWith2Thresholds, Suppress* e *Calendar*); Comportamento depois da correspondência (*TakeNext* e *DontCont* devem ser especificadas como valor); Amostra e seu tipo (pattern =); Descrição de Evento; Acção e Contextos. Os tipos de regras suportadas pelo SEC, que são: *Single, SingleWithScript, SingleWithSuppress, Pair, PairWithWindow, SingleWithThreshold, SingleWith2Thresholds, Suppress* e *Calendar*. Os tipos de acções suportadas pelo SEC não foram desenvolvidos apenas para gerar eventos de saída, mas sim para fazer as regras interagir entre si através de contextos e eventos internos, mas também para guardar e gerir eventos internos e para ligar sistemas externos ao SEC. As regras seguintes são suportadas pelo SEC: *none, logonly, write, shellcmd, pipe, spawn, create, set, delete, add, report, event* e *reset*.

Por fim foi descrito a performance da aplicação SEC, sendo esta uma ferramenta com índices de exigência em termos de velocidade de CPU, muito baixos e que facilmente corre em plataformas antigas. O SEC corresponde muito bem, mesmo quando existem centenas de operações de correlação de eventos e contextos, simultâneamente activados e guardados na memória activa de trabalho. O programa SEC, consome menos de 5 MB de memória na maioria das arquitecturas, face a quantidade de eventos que consegue processar.

4. INTEGRAÇÃO DO *SISTEMA DE CORRELAÇÃO DE EVENTOS E NOTIFICAÇÕES* (SCEN) COM O SISTEMA NMSIS

4.1. INTRODUÇÃO AO SCEN

Depois da necessidade de estudo do sistema NMSIS e após a escolha da ferramenta SEC que vai acrescentar um novo motor de inferência ao sistema. Torna-se necessário descrever todo o ambiente de desenvolvimento que vai permitir instrumentar e dotar o sistema de gestão NMSIS com um novo bloco de correlação de eventos que não existia no passado. A introdução do um novo *Sistema de Correlação de Eventos e Notificações* (SCEN), é descrito no modelo da figura 15. O sistema SCEN juntamente com o NMSIS faz parte de um sistema integrado de gestão e correlação de eventos.

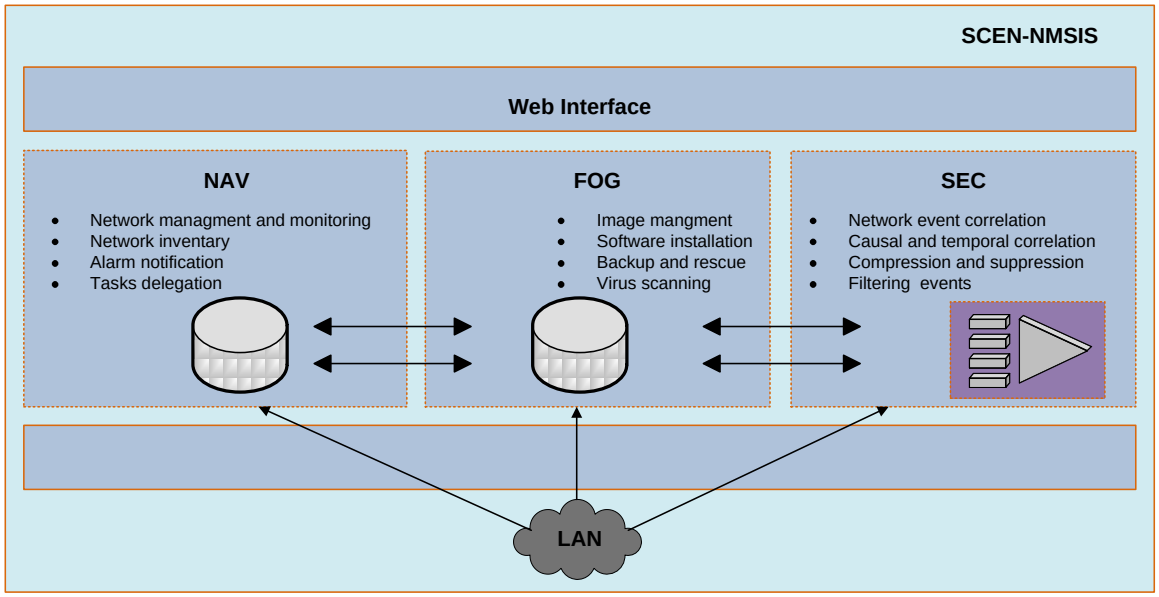


Figura 15 Ocorrência de quebras em vários segmentos de rede.

4.2. PLANTA DE TESTE NO LABORATÓRIO DE REDES (F507)

Para análise da integração da nova ferramenta *Sistema Correlação Eventos e Notificações* (SCEN) no sistema NMSIS, foi criado uma nova rede em ambiente de teste para que seja possível simular todos os eventos de entrada no sistema, sem ambiguidades e limpos de mensagens de sistemas externos. A tipologia desta rede é em forma de árvore, como demonstra a figura 16, assim é possível, simular quebras em qualquer segmento da rede e analisar a resposta do sistema às várias situações impostas. Com este protótipo de rede de testes é possível analisar o tipo de resposta que o sistema dá aos vários tipos de eventos de entrada.

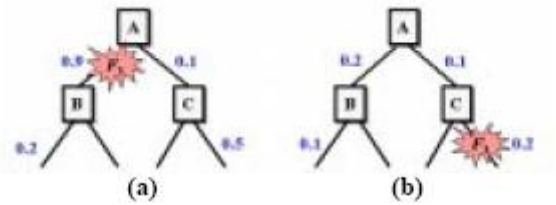


Figura 16 Ocorrência de quebras em vários segmentos de rede.

Para o efeito, foi criada uma nova rede, constituída por três Servidores, dois *Switchs* e um *Router*. Este tipo de cenário tem uma configuração que representa os principais elementos de uma rede real. O exemplo da figura 17, apresenta uma estrutura em árvore com duas redes distintas, ou seja, dois domínios de *broadcast* em layer 3 (camada de rede), cada ligação do router (Gateway) aos Switch é considerado um segmento de rede distinto. Neste contexto, existem ao todo cinco segmentos de rede onde poderão ser infligidos quebras físicas para simulação de eventos.

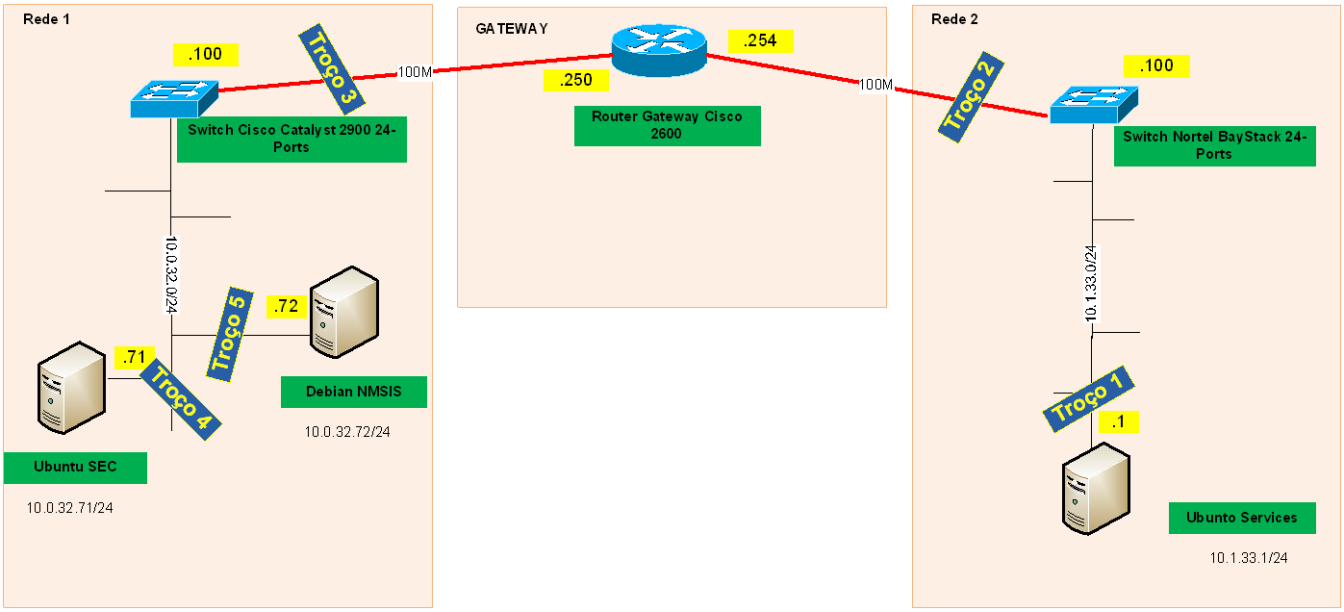


Figura 17 Esquema da rede de Teste na sala F507.

A rede é constituída, pelos seguintes elementos:

* Router: Cisco C2621;

* Switch da Rede 1- [10.0.32.100]: Cisco Catalyst 2900 de 24 Portos;

* Switch da Rede 2- [10.1.33.100]: Nortel Bay Stack de 24 Portos;

* Servido Debian- [10.0.32.72]: NMSIS;

* Servidor Ubuntu- [10.0.32.71]: NMSIS + SCEN;

* Servidor Ubuntu- [10.1.33.1]: Serviço http;

Foram configuradas duas redes distintas de classe C (256-2=254 endereços) para o efeito, a rede 1 foi criada com o endereço 10.0.32.24.0/24 e a rede 2 com o endereço 10.1.33.0/24. O router tem duas interfaces do tipo *Fast* Ethernet (F0/0 e F0/1) ao qual foi atribuído esses endereços de rede e respectivas máscaras. O endereço IP criado em cada uma dessas interfaces será o endereço de entrada e saída (*Default Gateway*) de cada rede. A rede 10.0.32.0 e o endereço de rede da *default gateway* 10.0.32.250 são os mesmos da rede real que actualmente são utilizados no departamento DEE. Assim não foi preciso alterar os endereços IP da máquina Debian NMSIS e da máquina Ubunto SCEN.

```
  <router>
interface Ethernet0/0
 description Rede 1 10.0.32.0
 ip address 10.0.32.250 255.255.255.0
 no ip directed-broadcast
 no cdp enable
!
interface Serial1/0:1
 description Rede 2 10.1.33.0
 ip address 10.1.33.254 255.255.255.0
 no ip directed-broadcast
```

O *switch* da Cisco foi configurado com endereço IP de gestão 10.0.32.100, através da Vlan 1, que é a Vlan de gestão nativa do *switch*.

```
  <swtich1>
interface VLAN1
 description gestao nativa switch1
 ip address 10.0.32.100 255.255.255.0
 no ip directed-broadcast
```

O *switch* da Nortel foi configurado com endereço IP de gestão 10.1.33.100, encapsulado pela *Vlan* 1, que é a gestão nativa do equipamento. A configuração deste *Switch*, não é em modo de comando mas sim em forma de menus, por esse motivo não foi apresentado neste documento a sua configuração.

4.3. MODELO DO SCEN

O *Sistema de Correlação de Eventos e Notificações* (SCEN) é constituído por quatro módulos:

- Correlador em Perl (SEC);
- Daemon em PHP faz as querys à NMSISdb;
- Daemon Net-SNMP (snmptrapd);
- Daemon em Perl que recebe as traps do Net-SNMP.

Este tipo de configuração do sistema permite analisar tanto, eventos recebidos do NMSIS via polling como eventos directamente da rede via Net-SNMP. O sistema SCEN está instalado na máquina Servidor Ubuntu- [10.0.32.71] junto com o NMSIS. O SCEN, tem vários *daemons* a correr em *background* que permitem fazer a comunicação entre as bases de dados do NMSIS e o SCEN. Este tipo de comunicação é feito por intermédio de *querys* às tabelas que tem informação de estado, dos elementos de rede e dos serviços de rede.

O SCEN através do *daemon* (*script* PHP), verifica e extrai toda a informação relevante que está guardada no NMSIS. A informação de estado dos elementos activos e dos serviços de rede que o NMSIS controla via *polling*, está concentrada num pequeno número de tabelas, o mesmo já não se passa se quizermos ter acesso à tipologia de rede. O NMSIS não dispõe de uma tabela que concentre informação de estado dos elementos em simultâneo com a topologia de rede, ou seja, ter informação do estado do elemento e informação dos elementos aos quais está ligado.

O NMSIS tem forma de saber os elementos que estão ligados a um determinado *Switch* ou os elementos que estão ligados a um determinado *Router*, mas não concentra essa informação num pequeno número de tabelas, essa informação encontra-se dispersa. A sua plataforma WEB disponibiliza esses dados visualmente ao utilizador, mas essa informação está distribuída por diversas tabelas, segundo um modelo próprio. Cabe ao SCEN juntar toda essa informaçao na forma de um mapa topológico de rede juntando a informação de estado dos elementos e serviços. Essa tarefa é proposta através de um *daemon* programado em linguagem PHP, especificamente concebido para este trabalho, que corre em

background e faz a manipulação das tabelas e materializa os eventos com informação de estado e topologia da rede.

Por outro lado, o SCEN está preparado para receber e entregar ao seu motor de inferência, eventos provenientes directamente da rede através da configuração prévia dos elementos de rede (*Swich, Router*) para a capacidade de gerarem *traps*. As *traps,* são recolhidas e trabalhadas pelo *daemon snmptrapd* (Net-SNMP) e são guardadas num ficheiro de *log* ou numa *file stream*. Depois de enviadas para esse ficheiro existe um outro *daemon* programado em linguagem Perl a correr em *background*, com a função de colocar toda a informação das *traps* legível e capaz de ser verificada pelo correlador. Este sistema constituído pela ferramenta de correlação e os três *daemons* de comunicação são a arquitectura do SCEN, figura 18.

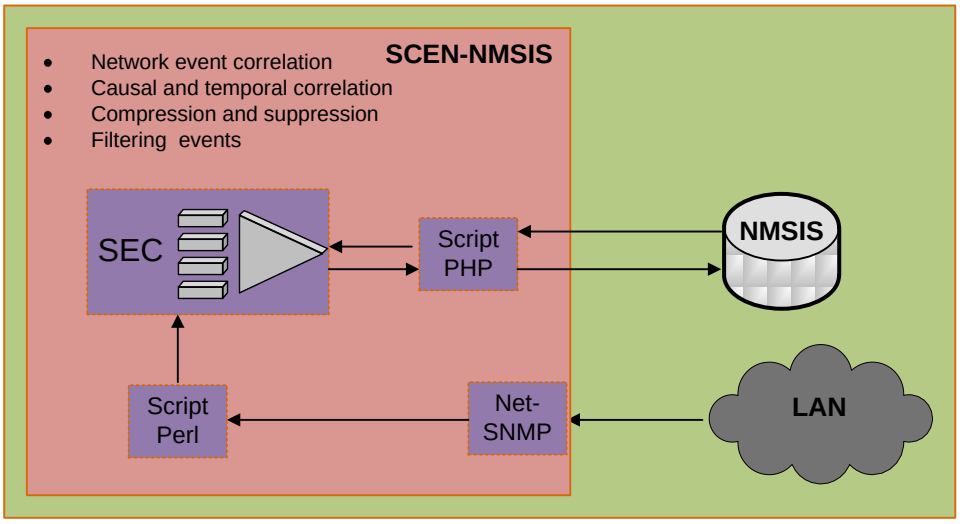


Figura 18 Arquitectura do protótipo SCEN

O SCEN tem dois tipos de eventos de entrada:

- Eventos de poolling, via *daemon* PHP, *querys* ao NMSISdb.
- Eventos de notificação (*traps*), via *daemon* Net-SNMP e *daemon* em Perl;

Para receber as notificações dos elementos de rede, procedeu-se à programação e configuração de três módulos. Passaremos à sua descrição.

4.4. SCRIPT DE LEITURA DO NMSISDB VIA QUERY – DAEMON PHP

Este script programado em linguagem *Hypertext Preprocessor* (PHP), funciona como um *daemon* que faz quatro tipos de *querys* (Query Serviço, Query SRV, Query SW, Query GW) à base de dados NMSISdb. Cada uma dessas *querys* seleciona e cruza vários tipos de tabelas com informação do serviço, dos elementos (box) e tabelas conjugadas de *routers* e *switchs*, ver descrição do NMSIS. O objectivo principal deste *script*, é a possibilidade de associar e aplicar à saída, através dos eventos, informação bastante essencial e relevante na identificação das falhas de rede. Desta forma os eventos terão na sua constituição vários parâmetros de informação relevante.

Analisaremos uns capítulos mais à frente, o que o *script* faz no seu todo em pormenor, quais as tabelas e quais os parâmetros que se pretende ver extraídos do NMSISdb, em cada caso. À sua entrada são dadas intruções via *query* de comandos SQL para a base de dados do NMSIS (NMSISdb) de forma a extrair e concactenar toda a informação. Na sua saída o *script* abre um ficheiro, faz *append* em cada linha de toda a informação concatenada em forma de eventos e fecha o ficheiro. Senão conseguir abrir ou não conseguir escrever, envia mensagens de aviso. O ficheiro de saída será um ficheiro de entrada normalizado do tipo *First In First Out* (FIFO), que será utilizado como ficheiro de *buffer* ou canal de entrada no sistema de correlação.

Nos capítulos posteriores será feita uma análise com informação mais detalhada de como este *script* comunica com as bases de dados e como faz o cruzamento de informação de forma a desenvolver quatro tipos de eventos de saída. E de que forma os quatro tipos de eventos no seu conjunto permitem recriar um mapa topolológico de toda a rede, com informação de estado de todos os activos e qual os interfaces dos elementos remotos aos quais estão ligados.

Exemplo da *query* de *Swich* (Query SW) com selecção de quatro tabelas (*netbox, arp, gwportprefix* e *gwport*):

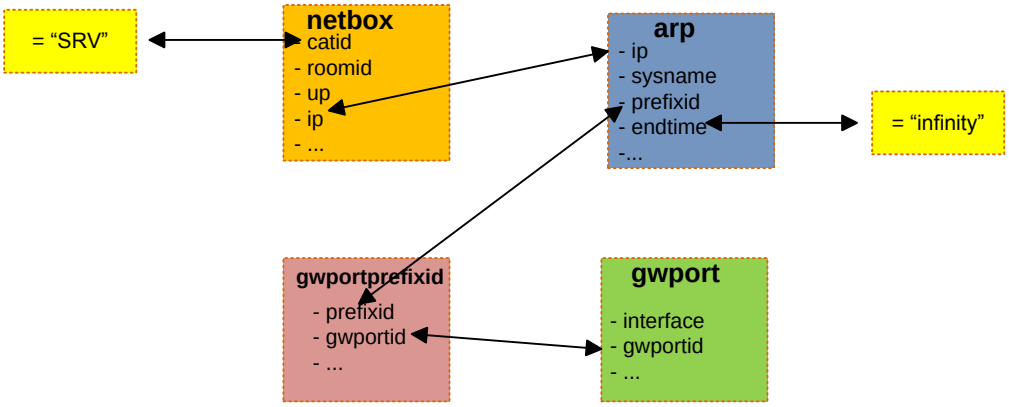


Figura 19 Cruzamento das tabelas para criação de evento do tipo query (Query SW)

Exemplo de evento de saída:

```
Query SW IP: arp.ip SalaID: netbox.catid Estado: netbox.up
CONECTADO Elemento: arp.sysname Porto: gwport.interface
```

Colocando agora o valor das variáveis no evento de saída:

```
Query SW IP: 10.1.33.100 SalaID: F507 Estado: n CONECTADO
Elemento: 10.0.32.250 Porto: FastEthernet0/1
```

Podemos verificar e confirmar neste evento de saída que o SW com o IP 10.1.33.100, da sala F507, está com o estado em baixo (n), ou seja, está inacessível e está ligado ao elemento de rede com o IP 10.0.32.250 no porto “FastEthernet0/1” desse elemento. Com este tipo de eventos criados à saída do *script* é possível recriar um modelo topológico das ligações entre todos os elementos de rede. No próximo capítulo será novamente abordada esta questão, na criação dos eventos via *query*.

4.5. IMPLEMENTAÇÃO DO DAEMON NET-SNMP

O pacote de código fonte aberto *Net-SNMP* é uma implementação e um conjunto de aplicações, licenciados como Software Livre, que operam com o protocolo *Simple Network Management Protocol (*SNMP), cujo objectivo principal é a configuração e a monitorização de dispositivos [52]. Dentro deste pacote para além de termos programas que possibilitam a configuração do agente do sistema, como o *snmpd* que é um agente *Net-*

SNMP que fica à escuta num porto específico, por defeito no porto UDP 161, todos os interfaces com *IP versão 4* (IPv4) ficam à espera de pedidos do software de gestão SNMP nesse porto [102]. E após recepção do pedido, processa-o, colectando a informação pedida e/ou executando a operação do pedido e depois retorna a informação ao remetente. Existem outros programas como o *snmptrapd* que é uma aplicação Net-SNMP que recebe e armazena a informação (em *log*) de notificação (SNMP TRAP) e mensagens de informação (INFORM).

Esta ferramenta foi integrada no nosso sistema para fazer a tradução das *traps* pelo módulo *snmptrapd* revelou-se de grande utilidade, pois contribuiu para a colecta de *traps* vindo dos equipamentos de rede de uma forma mais simples e célere. Por defeito está sempre à escuta em todos os interfaces com IPv4 no porto UDP 162. O *snmptrapd* deve ser corrido como *root*. Como o sistema Linux *Ubuntu* já tem o pacote/ferramenta *Net-SNMP* instalado por defeito não foi preciso fazer a sua instalação. Para que fosse possível activar um *daemon* que faça o tratamento das *traps* SNMP, foi configurado o agente *snmpd* para não ficar desactivado [100], [102], e para isso foi editado o etc/default/snmpd, que controla actividade do *snmpd* e do *snmptrapd*, da seguinte forma para que o agente interno *snmpd* fique inactivo:

```
#snmpd control (yes means start daemon)
SNMPDRUN=no
```

Para que seja possível activar o *daemon snmptrpd*, foi editado o etc/default/snmp da seguinte forma:

```
#snmptrapd control (yes means start daemon)
TRAPDRUN=yes
```

Assim foi possível activar o *daemon snmptrapd* para fazer a colecta e tratamento das *traps* enviadas pelos dispositivos de rede. O *snmptrapd* permite várias coisas como *traphandler* que activa uma aplicação após recepção de determinada *trap* com determinado OID. Permite também o tratamento de notificações de falha de autenticação. E o mais importante para este projecto, fazer armazenamento das notificações (*trap log*).

O ficheiro de configuração do receptor de notificações *Net-SNMP* é o *snmptrapd.conf*, este ficheiro usa um ou mais ficheiros de configuração para permitir o controlo da sua operação e como as *traps* podem ser processadas. Para desabilitar o tratamento de "*authentication failure traps* ", foi configurado *etc/snmp/snmptrapd.conf*, da seguinte forma [99], [102], [103]:

```
# ignoreauthfailure: Ignore authentication failure traps
# arguments: (1|yes|true|0|no|false)
ignoreauthfailure yes
disableAuthorization yes
```

Para activar o armazenamento de traps:

```
# donotlogtraps: Prevent traps from being logged
# Useful when you only want to use traphandles arguments:
# (1|yes|true|0|no|false)
donotlogtraps false
```

Escolher o ficheiro de armazenamento (log):

```
# logoption: Set options controlling where to log to See
# -L options in the snmptrapd.conf man page
logOption f /var/log/traplog
```

A função que interessa ao projecto é poder fazer a colecta e o tratamento das *traps* pelo *snmptrapd* e posteriormente canalizar as *traps* através do daemon Perl, para um ficheiro de entrada normalizado do tipo *fifo*, que vai servir de input ao sistema de correlação. Após concluída toda a configuração é necessário activar todas estas funções anteriores, através do *stop/start* do *snmpd* contido no directório *init.d*.

4.6. SCRIPT PERL DE INPUT DE TRAPS VIA NET-SNMP

Foi programado um *script* em linguagem Perl, com o objectivo principal de funcionar como um *daemon* que corre em plano de fundo. Este *daemon* faz o processamento de todas as linhas de informação que pertencem a uma *trap* recebida do s*nmptrapd* (Net-SNMP) e coloca toda a informação numa única linha para ser trabalhada pelo motor de inferência da ferramenta de correlação. Já que esse motor faz reconhecimento do padrão definido, linha-a-linha durante o tempo de execução. A sua entrada e a sua saída são ficheiros ou *file*

streams do tipo FIFO. A sua entrada são os eventos de *logging* da saída do *snmptrapd* e a sua saída é a entrada do sistema de correlação.

4.6.1. CONFIGURAÇÃO DE TRAPS NOS EQUIPAMENTOS DE REDE

Como já foi demonstrado, existem programas específicos que estão à escuta em portos específicos do protocolo de transporte UDP, que pertence à camada 4 do modelo OSI. Esses módulos fazem a recepção e tratamento das notificações (traps) que são enviadas em tempo real pelos activos de rede.

Sempre que é necessário que os elementos de rede enviem notificações (*traps*) de alarmes, terá que se activar essa geração nos dispositivos, através da configuração de certos parâmetros. Escolhe-se o tipo de notificações que se pretende ver activados nos equipamentos e os nomes das *comunity string* que serão geradas nas *traps*.

É preciso configurar o endereço de rede do servidor de destino que lógicamente será a máquina com o *daemon* Net-SNMP activo, para onde serão enviadas as *traps* via porto 162 através do protocolo de camada de transporte UDP, não orientado às conexões. As notificações (*traps*) serão enviadas em tempo real pelos activos de rede, para isso foi preciso configurar o *Router* e os dois *Swtichs* de rede para gerarem *traps*.

Exemplo da configuração no Router:

```
snmp-server enable traps snmp
snmp-server enable traps config
snmp-server enable traps entity
snmp-server enable traps envmon
snmp-server enable traps syslog
snmp-server host 10.0.32.71 traps public
```

4.6.1. CONSIDERAÇÕES FINAIS

Como se pode ver na figura 20, o sistema SCEN tem a possibilidade de recolher eventos extraídos da base de dados NMSISdb e eventos de notificações (*traps*) da rede. Analisaremos nos capítulos posteriores, em maior pormenor, como essa informação será tratada no correlador de eventos.

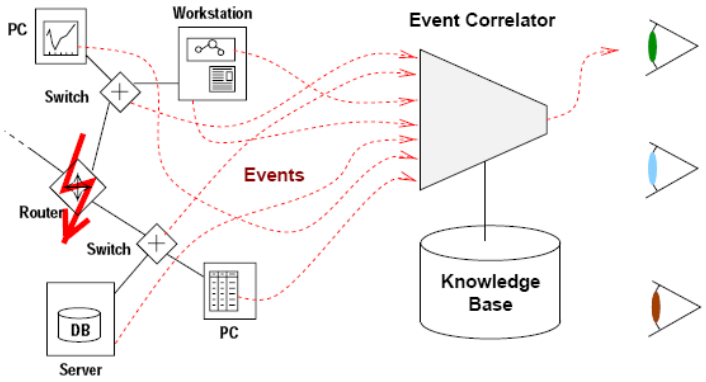


Figura 20 SCEN recebe eventos via *pooling* da DB e *traps* dos elementos de rede [127].

5. EVENTOS E SISTEMA DE REGRAS E ACÇÕES DE CORRELAÇÃO

5.1. INTRODUÇÃO AOS TIPOS DE EVENTOS DO SCEN

Como já foi comentado, o SCEN tem duas formas de comunicar com o exterior, uma delas é por intermédio de um *daemon* especialmente concebido neste projecto e que faz *querys* ao NMSIS. A outra forma é através de dois *daemons*, um criado para este projecto e o outro programado para serem parte integrante do sistema. Estes últimos recebem as traps directamente dos elementos de rede. A figura 21, mostra os dois tipos de comunicação do SCEN (via *query* e via *trap*).

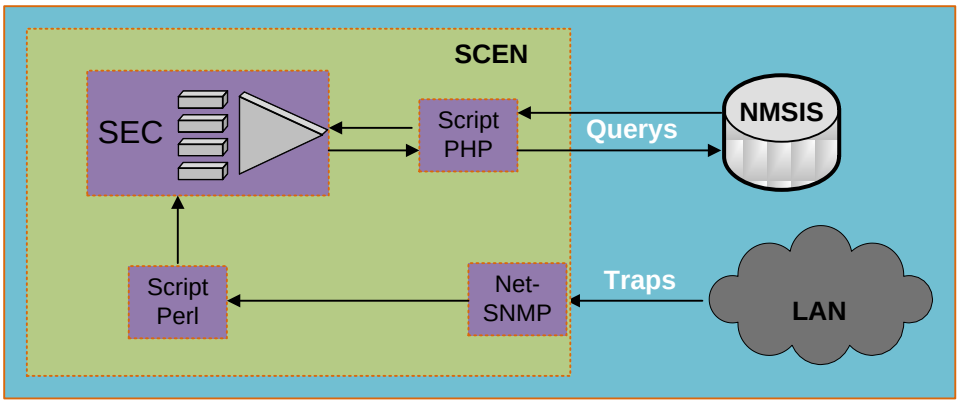


Figura 21 Arquitectura do protótipo SCEN

A informação que circula nas vias de comunicação, são eventos. Quando o SCEN por intermédio dos seus *daemons* faz *querys* as bases de dados do NMSIS os eventos recebidos e entregues ao motor de inferência são considerados eventos do tipo "**Query**". Ou seja o SCEN, não faz o polling aos elementos de rede, esse trabalho de *front-end* cabe às ferramentas internas e específicas do NMSIS. O SCEN procura e trabalha a informação guardada na base de dados do NMSIS.

Quando o SCEN por intermédio dos seus *daemons* recebe e trabalha a informação contida nas *traps*, enviadas pelos activos de rede (*Switchs*, *Routers*) esses eventos de rede são considerados eventos do tipo "**traps**". Ou seja, a informação que circula nessas vias de comunicação são notificações geradas instantâneamente pelos activos de rede.

5.2. TIPOS DE EVENTOS DE POOLING (VIA *QUERY AO NMSISDB)*

Os eventos do tipo *query* são obtidos, através do cruzamento de parâmetros de inúmeras tabelas do NMSISdb (Base Dados do NMSIS), através de um script em linguagem PHP, que faz quatro tipos de *query* às várias tabelas da base de dados. Dos quatro tipos de eventos de saída do *script* da base de dados NMSIS. Um dos eventos é relativo ao tipo de "Serviço" (Query Serviço). O segundo é relativo à informação dos elementos de rede do tipo "Servidor" (Query SRV). O terceiro é relativo à informação dos elementos de rede do tipo "Switch" (Query SW). Por fim, o quarto é relativo à informação dos elementos de rede do tipo "Gateway" (Query GW).

A conjugação das várias tabelas merece por isso uma atenção especial, já que é a base de sucesso deste modelo de eventos. Podemos começar a apresentar os quatro tipos de eventos

via *query* à base de dados NMSIS, e que são posteriormente entregues ao motor de correlação.

5.2.1. QUERY SERVIÇO

A *query* de eventos do tipo "Serviço" (Query Serviço) é feita através de duas tabelas, a tabela *netbox* e *service*. Da tabela netbox apenas se pretende obter o IP da máquina onde está o serviço e concatenar essa informação com o resto da informação contida na tabela service e criar o evento. A todos os eventos criados é feito appendá saída. Ver na tabela 6, o conteúdo do evento "Query Serviço ".

Tabela 6 Descrição dos campos do evento "Query Serviço"

Endereço IP	Endereço do elemento de rede
Serviço	Identificação do tipo de serviço que está ser feito *pooling* pelo NMSIS
Gravidade	Acessível ou não acessível (y, n)

Entrada:

```
SELECT netbox.ip AS ip1,service.handler AS handler2,service.up
AS up2  FROM netbox, service WHERE netbox.netboxid =
service.netboxid
```

Exemplo de evento de saída:

```
Query Serviço IP: 10.0.32.71 TipoServiço: http Estado: n
```

5.2.2. QUERY SERVIDOR (SRV)

A *query* de eventos de elemento de rede "Servidor" (Query SRV) é feita através de três tabelas *netbox*, *arp* e *cam*. Pretende-se através da tabela *netbox* retirar os valores do endereço IP, da localidade e estado do elemento Servidor que esteja ligado ao primeiro ponto de acesso da rede, neste caso um switch. É feito o cruzamento do mesmo IP entre a tabela *netbox* e da tabela *arp*. A tabela de *arp* tem os valores da tabela *Address Resolution Protocol* (ARP) do router gateway, tabela esta, que tem a informação de todos os endereços IP (camada 3 do modelo OSI) e respectiva correspondência em endereços *Media*

Access Control (MAC) da camada 2 do modelo OSI, ainda activos (através do valor da variável igual a "*infinity*" da tabela *arp*).

Desta forma, permite-nos um cruzamento muito útil para se obter o porto e o elemento de destino ao qual está ligado o elemento Servidor (que será certamente um switch). O porto e o elemento de destino ao qual está ligado o elemento Servidor através do MAC da tabela *arp* puderá ser obtido através do cruzamento com a tabela *cam* que é a tabela de correspondência dos portos de todos os switchs na rede (e outras informações não retiradas com relevância ao nível do porto do switch), aos respectivos endereços MAC conectados. Toda esta informação é concatenada e colada no momento de criação do evento, poderemos ver na tabela 7, os campos do evento "Query Servidor"..

Tabela 7 Descrição dos campos do evento "Query Servidor"

Categoria do Elemento	Identificação do tipo de elemento de rede que está ser feito pooling pelo NMSIS. Servidor (SRV), Gateway (GW), Switch (SW), outro tipo (OTHER)
Endereço IP	Endereço IP do elemento de rede (servidor)
Local	Identificação da sala
Gravidade	Acessível ou não acessível (y, n)
Endereço IP	Endereço IP do elemento de rede remoto (switch)
Porto	Identificação do elemento de rede remoto (switch)

Entrada:

```
SELECT netbox.netboxid AS netboxid1,netbox.catid AS
catid1,netbox.roomid AS roomid1,netbox.up AS up1,arp.ip AS
ip2,arp.mac AS mac2,cam.sysname AS sysname3,cam.port AS port3
FROM netbox, arp, cam WHERE arp.mac=cam.mac AND netbox.ip =
arp.ip AND arp.end_time ='infinity'
```

Exemplo de evento de saída:

```
Query SRV IP: 10.1.33.1 SalaID: F507 Estado: y CONECTADO
Elemento: 10.1.33.100 Porto: BayStack 450-24T - 1
```

5.2.3. QUERY SWITCH (SW)

A *query* de eventos de elemento de rede "Switch" (Query SW) é feita através de quatro tabelas *netbox*, *arp*, *gwportprefix* e *gwport*. Pretende-se através da tabela *netbox* retirar os valores do endereço IP, da localidade e estado do elemento Switch que esteja ligado a um ponto de rede camada 3 neste caso um router. É feito o cruzamento do mesmo IP entre a tabela *netbox* e da tabela *arp*. A tabela de *arp* tem os valores da tabela *Address Resolution Protocol* (ARP) do router gateway, tabela esta, que tem a informação de todos os endereços IP (camada 3 do modelo OSI) e respectiva correspondência em endereços *Media Access Control* (MAC) da camada 2 do modelo OSI, ainda activos (através do valor da variável igual a "*infinity*" da tabela *arp*).

Da tabela *arp* é retirado o *sysname* que é o identificador do elemento de rede remoto (Gateway) ao qual o Switch está conectado. É efectuado o cruzamento das tabelas *arp* com *gwportprefix* e *gwport* para se obter o porto do elemento de destino ao qual está ligado o elemento *Switch* (que será certamente um Gateway). Toda esta informação deve ser colada ao evento, poderemos ver na tabela 8, os campos do evento.

Tabela 8 Descrição dos campos do evento "Query Switch"

Categoria do Elemento	Identificação do tipo de elemento de rede que está ser feito pooling pelo NMSIS. Servidor (SRV), Gateway (GW), Switch (SW), outro tipo (OTHER)
Endereço IP	Endereço IP do elemento de rede (switch)
Local	Identificação da sala
Gravidade	Acessível ou não acessível (y, n)
Endereço IP	Endereço IP do elemento de rede remoto (gateway)
Porto	Identificação do elemento de rede remoto (gateway)

Entrada:

```
SELECT netbox.catid AS catid1,netbox.roomid AS
roomid1,netbox.up AS up1,arp.ip AS ip2,arp.sysname AS
sysname2,gwport.interface AS interface4 FROM netbox, arp,
gwportprefix, gwport WHERE netbox.ip = arp.ip AND arp.prefixid
= gwportprefix.prefixid AND gwportprefix.gwportid =
gwport.gwportid AND arp.end_time ='infinity' AND netbox.catid
='SW'
```

Exemplo de evento de saída:

```
Query SW IP: 10.1.33.100 SalaID: F507 Estado: n CONECTADO
Elemento: 10.0.32.250 Porto: FastEthernet0/1
```

5.2.4. QUERY GATEWAY (GW)

A *query* de eventos de elemento de rede "Gateway" (Query GW) é feita apenas à tabela *netbox*. Pretende-se através da tabela *netbox* retirar os valores do endereço IP, da localidade e do estado do elemento Gateway, tabela 9.

Tabela 9 Descrição dos campos do evento "Query Gateway"

Categoria do Elemento	Identificação do tipo de elemento de rede que está ser feito pooling pelo NMSIS. Servidor (SRV), Gateway (GW), Switch (SW), outro tipo (OTHER)
Endereço IP	Endereço IP do elemento de rede (switch)
Local	Identificação da sala
Gravidade	Acessível ou não acessível (y, n)

Entrada:

```
SELECT netbox.catid,netbox.ip,netbox.roomid,netbox.up FROM
netbox WHERE netbox.catid ='GW'
```

Exemplo de evento de saída:

```
Query GW IP: 10.0.32.250 SalaID: F507 Estado: n
```

5.3. TIPOS DE EVENTOS DE *TRAPS* (VIA DAEMON NET-SNMP)

Como foi demonstrado nos capítulos anteriores, as notificações (*traps)* que são enviadas pelos equipamentos activos de rede, são recolhidas pelo *deamon* "*snmptrapd*" do *Net-SNMP* que será encarregue de fazer o *log* dessas *traps* num *pipe* (ficheiro tipo *fifo*).

Exemplo de um *trap* de um *switch*:

```
2010-09-01 01:28:47 10.0.32.100(via UDP: [10.0.32.100]:54428) TRAP, SNMP v1,
community public SNMPv2-SMI::enterprises.9.1.217 Link Down Trap (0) Uptime: 32
days, 7:47:32.86 IF-MIB::ifIndex.25 = INTEGER: 25IF-MIB::ifDescr.25 = STRING:
FastEthernet0/24 IF-MIB::ifType.25 = INTEGER: ethernetCsmacd(6)SNMPv2-
SMI::enterprises.9.2.2.1.1.20.25 = STRING: "down"
```

As *traps*, tem inúmera informação, toda esta informação passa por um módulo que corre em linguagem Perl com a tarefa de processar todas as linhas de informação de cada uma das *traps* recebidas do "*snmptrapd*" e colocar cada *trap* numa única linha, para a correspondência padrão executada pelo motor de inferência do SEC, que faz o varrimento linha-a-linha durante o tempo de execução. De toda esta informação, a que interessa realmente utilizar é a seguinte:

Tabela 10 Descrição dos campos da Notificação de Alarme via *trap*

Data	Data do Relatório de Alarme no formato: AAAA-MM-DD
Hora	Hora do Relatório de Alarme no formato: HH:MM:SS
Endereço IP	Endereço IP do elemento de rede que envia a *trap* (xxx.xxx.xxx.xxx)
Interface	Identificação da Interface (Porto)
Gravidade	up, down, Link up trap, link down trap, Keepalive failed, Keepalive OK

5.4. ESTUDO DA CORRELAÇÃO DE CAUSALIDADE DO SCEN

Como referido nos capítulos anteriores e com base no tipo de eventos que entram no sistema SCEN. Será apresentado um estudo de como os eventos podem ter relações de causalidade (causa-efeito) entre eles. Serão feitos testes sistemáticos ao eventos produzidos devido as quebras físicas infligidas nos segmentos da rede de teste, para que o SCEN possa dar uma resposta muito aproximada na descoberta da raiz da causa do problema.

De acordo com o esquema da figura 17 e devido a determinadas circunstâncias que possam ocorrer na rede, como falhas nos segmentos de rede (1, 2, 3, 4 e 5). Foram estabelecidos vários princípios de acordo com os eventos que foram sendo produzidos através de sistemáticas falhas nos segmentos de rede. Os eventos que são recebidos via *query* são os seguintes, foram atribuidos símbolos em vez de valores para representar cada uma das variáveis:

- `Query Serviço IP: λ Serviço: Y Estado: τ`
- `Query θ IP: λ SalaID: α Estado: μ CONECTADO Elemento: ζ Porto: σ`
- `Query θ IP: λ SalaID: α Estado: μ CONECTADO Elemento: ρ Porto: φ`
- `Query θ IP: λ SalaID: α Estado: μ`

As notificações (*traps*) têm também variáveis que interessam ser processadas. Os eventos que são recebidos via *trap*, são do tipo:

- `Trap IP: α Interface: η Estado: π`

Algumas destas variáveis, seja dos eventos via *query*, seja dos eventos via *trap* mudam a cada instante no tempo, representando vários valores. A tabela 11 representa quais os valores que podem assumir essas variáveis, bem como a origem de onde provêm.

Tabela 11 Descrição dos valores que podem ser atribuídos às variáveis

	CAMPO	VARIÁVEL	*TABELA*	*COLUNA*	*MIB*	VALORES
Query Servidor	IP	λ	*netbox*	*ip*		xxx.xxx.xxx.xxx
	Serviço	Y	*service*	*handler*		http, ftp, etc
	Estado	τ	*service*	*up*		y, n
Query Elementos Rede	Query [Categoria]	θ	*netbox*	*catid*		SRV, GW, SW, OTHER
	IP	λ	*netbox*	*ip*		xxx.xxx.xxx.xxx
	SalaID	α	*netbox*	*roomid*		Salas DEE (F101, …, F520)
	Estado	μ	*netbox*	*up*		y, n
	Elemento Remoto	ζ	*cam*	*sysname*		xxx.xxx.xxx.xxx
	Porto	σ	*cam*	*port*		A-Za-z0-9!@#$%^&*
	Elemento Remoto	ρ	*arp*	*sysname*		xxx.xxx.xxx.xxx
	Porto	ϕ	*gwport*	*interface*		A-Za-z0-9!@#$%^&*
trap	Net-SNMP => (ex:IP)	α			ip	xxx.xxx.xxx.xxx
	Net-SNMP => (ex:Interface)	η			interface	Número Inteiro (1, 2 ,.., n)
	Net-SNMP => (ex:Estado)	π			state	up, down, Link up trap, link down trap, Keepalive failed, Keepalive OK;

Vamos analisar cada um dos valores que pode ser atribuído às variáveis para formar um conjunto de eventos que interessam ser processados e relacionadas as suas interacções. Para isso é preciso ter uma amostra dos tipos de eventos que o sistema SCEN pode ter à sua entrada.

Os estados (y e n) representam respectivamente o sucesso ou insucesso do *pooling* feito pela ferramenta NMSIS aos elementos de rede e também a determinados serviços. Se o *pooling* não falhar o estado da variável será "y", se falhar o estado será "n". Nas *traps* o valor é perceptível para as saber distinguir. O mesmo se passa com eventos de notificação em que o estado de determinado elemento é mais descritivo (*link up trap, link down trap, Keepalive failed, Keepalive failed* e *Keepalive OK*).

Query de serviço:

- Query Serviço IP: λ Serviço: Y Estado: **y** (evento 0)
- Query Serviço IP: λ Serviço: Y Estado: **n** (evento 1)

Query Elemento Rede Servidor (SRV):

- Query θ IP: λ SalaID: α Estado: **y** CONECTADO Elemento: ζ Porto: σ (evento 2)
- Query θ IP: λ SalaID: α Estado: **n** CONECTADO Elemento: ζ Porto: σ (evento 3)

Query Elemento Rede Switch (SW):

- Query θ IP: λ SalaID: α Estado: **y** CONECTADO Elemento: ρ Porto: φ (evento 4)
- Query θ IP: λ SalaID: α Estado: **n** CONECTADO Elemento: ρ Porto: φ (evento 5)

Query Elemento Rede Gateway (GW):

- Query θ IP: λ SalaID: α Estado: **y** (evento 6)
- Query θ IP: λ SalaID: α Estado: **n** (evento 7)

Trap Elemento Rede Switch:

- Trap IP: α Interface: η Estado: **link up trap** (evento 8)
- Trap IP: α Interface: η Estado: **link down trap** (evento 9)

Trap Elemento Rede Gateway:

- Trap IP: α Interface: η Estado: **Keepalive OK** (evento 10)
- Trap IP: α Interface: η Estado: **Keepalive failed** (evento 11)

Cada um destes eventos tem uma identificação (0-11), cada ocorrência de determinado evento tem uma causalidade. Considerando o esquema da figura 17 (Esquema da rede de Teste na sala F507), dependendo das falhas físicas aplicadas a cada um dos segmentos de rede, podemos assumir determinados factos e pressupostos.

Nesse sentido, foi construído um mapa de causalidade na figura 22. Este mapa em árvore mostra a relação de causa efeito entre eventos recebidos via *query* (à esquerda) e o conjunto dos dois via query e via *trap* (à direita).

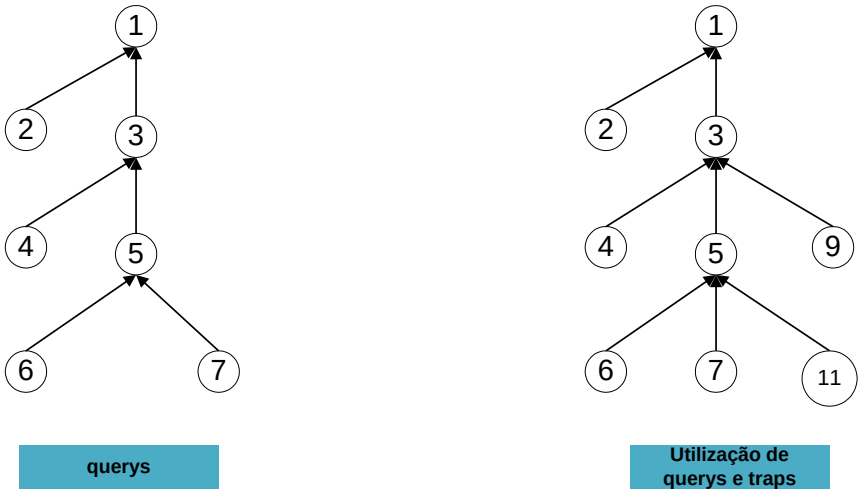


Figura 22 Mapa em árvore da relação de correlação de causalidade das *querys* e *traps* e o conjunto das duas

Uma das aproximações mais comuns utilizadas hoje em dia, são as correlações à base de regras. Cada regra tem a seguinte forma: [condição se (*IF*) então (*THEN*) acção]. No caso de motores de correlação à base de regras, o processo de correlação de eventos é totalmente personalizável e determinado por quem as faz as regras [123]. Esta abordagem, é, a mais aceite actualmente por entre os engenheiros de gestão de redes.

Existem diversos tipos de relações entre os eventos reproduzidos nos testes de laboratório, de acordo com o esquema de rede montado na figura 17. As relações entre os eventos são as seguintes:

Se analisarmos só as *querys*, esquema da esquerda da figura 22:

- Se existir um a quebra no serviço (evento 1), então a sua causa pode estar relacionada com o evento de falha no servidor (evento 3) ou não (evento 2).
- Se existir uma quebra no elemento de rede servidor (evento 3), então pode estar relacionada com falha no *switch* (evento 5) ou não (evento 4).
- Se existir uma quebra no elemento de rede switch (evento 5), então pode estar relacionada com falha na *gateway* (evento 7) ou não (evento 6).

Se juntarmos mais *traps* com a mesma informação, esquema da direita:

- Se existir um a quebra no serviço (evento 1), então a sua causa pode estar relacionada com o evento de falha no servidor (evento 3) ou não (evento 2).
- Se existir uma quebra no elemento de rede servidor (evento 3), então pode estar relacionada com falha no *switch* (evento 5) ou não (evento 4 ou 9).
- Se existir uma quebra no elemento de rede switch (evento 5), então pode estar relacionada com falha na *gateway* (evento 7) ou não (evento 6 ou 11).

A partir desta análise foi desenvolvido um algoritmo de correlação entre os eventos observados. Após recepção destes eventos à entrada, o motor de correlação com base no algoritmo proposto, correlaciona-os e inferindo-lhes uma causa para o tipo de efeitos processados. Os efeitos são os eventos analisados na árvore e a causa advém de outros eventos. As causas também podem ser inferidas através da criação de novos eventos, este é um dos pressupostos do protótipo de algoritmo que se pretende criar neste trabalho.

Os novos eventos criados são do tipo: alarmes que surgem de falhas de serviços " *falha na conexão ao serviço"*, falhas nos elementos (servidor, *gateway* ou *switch)* e o evento é " *falha na conexão ao [Servidor, Gateway, Switch]"*, podem ser atribuídas a falhas nos interfaces das gateways " *falha na ligação da Gateway"* e falhas nos interfaces dos *switchs* " *falha na ligação do Switch"*. Será presentado na figura 23, a estrutura em esqueleto do estudo da rede de teste com o mapa de causalidade dos eventos de rede inferidos por correlação.

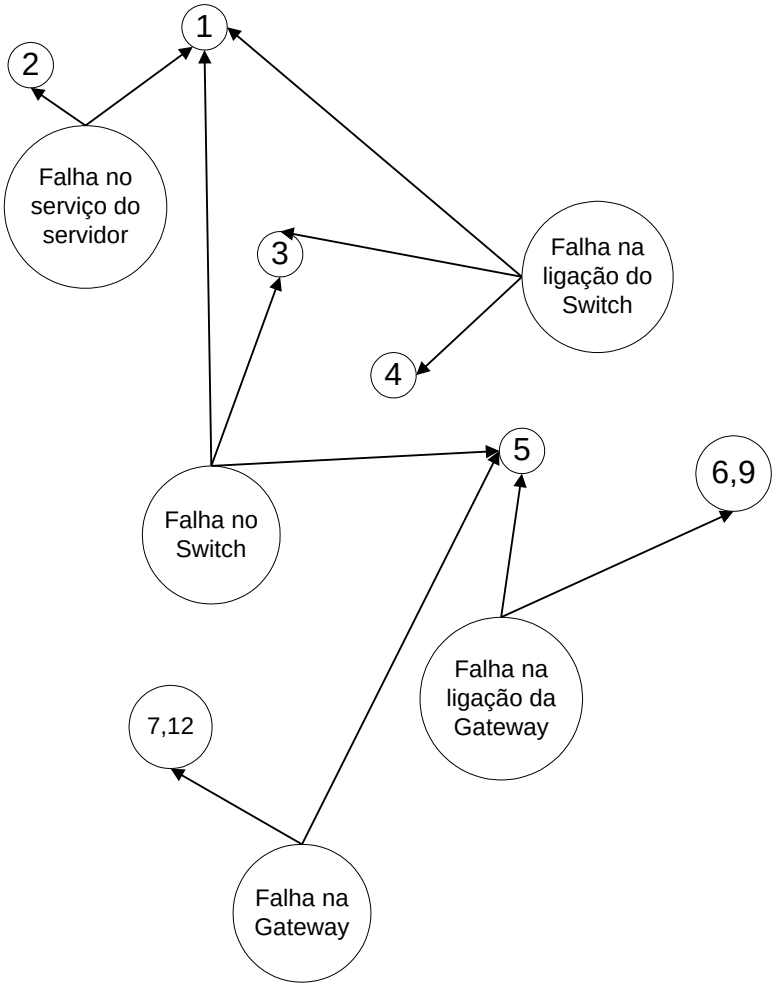


Figura 23 Mapa em árvore da relação de correlação de causalidade das *querys* e *traps*. Novos eventos de rede são inferidos por correlação entre estes eventos.

Recorrendo à observação das relações entre os eventos e respectiva formação de novos eventos, podemos chegar ao objectivo proposto por este trabalho que é "Qual a raiz da causa do problema", para esta pergunta segundo o modelo proposto pelo algoritmo

podemos ter a resposta. A figura 23, traduz a relação de causalidade da figura 22, com a representação dos efeitos e as causas que são inferidas através da correlação entre eventos e criação de novos eventos, de acordo com as seguintes condições:

a. Se existir uma quebra no serviço (evento 1) e não existir uma quebra no elemento de rede servidor (evento 2) então cria evento "*falha no serviço Y do servidor λ"*.

b. Se existir uma quebra no serviço (evento 1) e existir uma quebra no elemento de rede servidor (evento 3) e não existir uma quebra no elemento de rede *switch* (evento 4) então cria evento "*falha no serviço Y do servidor λ devido a quebra na ligação entre servidor λ e switch ρ porto σ"*.

c. Se existir uma quebra no serviço (evento 1) e existir uma quebra no elemento de rede servidor (evento 2) e existir uma quebra no elemento de rede *switch* (evento 5) e não existir uma quebra no elemento de rede gateway (evento 6 ou 9) então cria evento "*falha no serviço Y do servidor λ devido a quebra na ligação entre switch λ e gateway ζ porto φ "*.

d. Se existir uma quebra no serviço (evento 1) e existir uma quebra no elemento de rede servidor (evento 2) e existir uma quebra no elemento de rede *switch* (evento 5) e existir uma quebra no elemento de rede gateway (evento 7 ou 11) então cria evento "*falha no serviço Y do servidor λ devido a Gateway λ isolada"*.

Adaptando estas notificações de saída do sistema SCEN ao esquema da figura 23, podemos fazer determinadas afirmações que visam a descoberta da raiz da causa do problema de rede.

Se o serviço por alguma razão falhar mas o servidor estiver activo e acessível temos um único evento a reportar o seguinte: "*falha no serviço Y do servidor λ"*.

Se o troço 1, 4 ou 5 ficar cortado por alguma razão teremos, outro, e único evento a reportar "*falha no serviço Y devido a quebra na ligação entre servidor λ e switch ρ porto σ "*.

Se por qualquer outra causa o troço 2 ou 3 ficar cortado teremos também apenas um evento a reportar "*falha no serviço Y devido a quebra na ligação entre switch* ρ *e gateway ζ porto φ"*.

Por fim, se a Gateway estiver inacessível por alguma razão teremos um único evento a reportar "*falha no serviço Y do servidor λ devido a Gateway λ isolada"*.

Todos os eventos recebidos no sistema via *query* serão concentrados num só e único evento de saída do sistema. Este é o objectivo principal deste sistema de correlação, poderemos ter inúmeros eventos de *query*, seja de serviço ou de estado dos elementos, mas teremos no final à saída do correlador, uma única causa resumida e raíz principal de todos os eventos, imagine-se vários eventos de entrada devido à falha de um Switch ou de um Router Gateway da rede.

5.5. ESTUDO DA CORRELAÇÃO TEMPORAL DO SCEN

Nos capítulos anteriores foi abordado a questão da relação entre eventos com base no tempo. No caso de correlação causal existe uma série de eventos identificados pelo tipo de elemento de rede e o seu estado, no instante em que deu entrada no sistema SCEN. Para que se faça uso da correlação temporal, para além da identificação do tipo de elemento de rede e o seu estado é preciso informação adicional acerca do instante temporal em que ocorreu. Esta relação é para ser feita durante o histórico dos eventos, com uma amostragem temporal dos instantes de ocorrência. Para alcançar este fim, foi elaborado um conjunto de procedimentos para serem integrados este tipo de correlações temporais entre os eventos no sistema SCEN.

No caso de eventos via *query* à base dados *PostgreSQL* do NMSIS, como é feita a cada 30 segundos, que é o tempo de actualização via *pooling* da base de dados do NMSIS. Não fazia sentido este tipo de correlação com base no tempo. Embora pudesse ser anexada a informação da hora de chegada do evento à entrada do SCEN. Foi entendido que seria de maior utilidade utilizar este mecanismo para eventos que são gerados instantâneamente por notificação (*traps*) dos equipamentos activos de rede (*Routers*, *Switchs*, etc). Os tipos de *traps* que aparecem no sistema SCEN são analisados do ponto de vista temporal por ordem de chegada ao sistema SCEN, a sua hora de origem pode também ser trabalhada de forma a sabermos a hora exacta que aconteceu no equipamento de origem. Exemplo de *traps* que contêm o instante da sua ocorrência:

Traps do switch Cisco Catalyst C2900:

Down (Interface em baixo):

2010-09-01 01:28:47 10.0.32.100(via UDP: [10.0.32.100]:54428) TRAP, SNMP v1, community public SNMPv2-SMI::enterprises.9.1.217 Link Down Trap (0) Uptime: 32 days, 7:47:32.86 IF-MIB::ifIndex.25 = INTEGER: 25IF-MIB::ifDescr.25 = STRING: **FastEthernet0/24** IF-MIB::ifType.25 = INTEGER: ethernetCsmacd(6)SNMPv2-SMI::enterprises.9.2.2.1.1.20.25 = STRING: **"down"**

Up (Interface em cima):

2010-09-01 01:45:50 10.0.32.100(via UDP: [10.0.32.100]:54428) TRAP, SNMP v1, community public SNMPv2-SMI::enterprises.9.1.217 Link Up Trap (0) Uptime: 32 days, 8:04:35.69 IF-MIB::ifIndex.25 = INTEGER: 25 IF-MIB::ifDescr.25 = STRING: **FastEthernet0/24** IF-MIB::ifType.25 = INTEGER: ethernetCsmacd(6) SNMPv2-SMI::enterprises.9.2.2.1.1.20.25 = STRING: **"up"**

Traps do switch Nortel Baystack:

Down (Interface em baixo):

2010-09-01 01:12:10 10.1.33.100(via UDP: [**10.1.33.100**]:161) TRAP, SNMP v1, community public SNMPv2-SMI::enterprises.45.3.35.1 **Link Down Trap** (0) Uptime: 32 days, 1:39:37.99 IF-MIB::ifIndex.1 = INTEGER: **1**

Up (Interface em cima):

2010-09-01 01:12:15 ***10.1.33.100*** **(**via UDP: [10.1.33.100]:161) TRAP, SNMP v1, community public SNMPv2-SMI::enterprises.45.3.35.1 **Link Up Trap** (0) Uptime: 32 days, 1:39:39.78 IF-MIB::ifIndex.1 = INTEGER: **1**

Traps do router Cisco C2600:

Down (Interface em baixo):

2010-09-01 01:46:24 0.0.0.0(via UDP: [**10.0.32.250**]:1464) TRAP, SNMP v1, community public SNMPv2-SMI::enterprises.9.1.209 Link Down Trap (0) Uptime: 5 days, 3:00:10.39 IF-MIB::ifIndex.1 = INTEGER: 1 IF-MIB::ifDescr.1 = STRING: **FastEthernet0/0** IF-MIB::ifType.1 = INTEGER: ethernetCsmacd(6) SNMPv2-SMI::enterprises.9.2.2.1.1.20.1 = STRING: **"Lost Carrier"**

Up (Interface em cima):

2010-09-01 01:46:34 0.0.0.0(via UDP: [**10.0.32.250**]:1464) TRAP, SNMP v1, community public SNMPv2-SMI::enterprises.9.1.209 Link Up Trap (0) Uptime: 5 days, 3:17:06.12 IF-MIB::ifIndex.1 = INTEGER: 1 IF-MIB::ifDescr.1 = STRING: **FastEthernet0/0** IF-MIB::ifType.1 = INTEGER: ethernetCsmacd(6) SNMPv2-SMI::enterprises.9.2.2.1.1.20.1 = STRING: **"Keepalive OK"**

As amostras são processadas pelo SCEN através de correspondência padrão e recorrendo às expressões regulares utilizadas na linguagem Perl. Essa correspondência padrão está a sombreado para melhor ser distinguida (*Bold*). Os campos dos eventos via *trap* serão processados por ordem de entrada no sistema e irá ser feita a sua relação de correlação temporal.

Um dos algoritmos utilizados para correlação temporal tem como objectivo principal diminuir o número de eventos que são lançados para o operador do sistema e de certa forma facilitar a sua compreensão do que está acontecer na rede a cada instante.

5.6. PERL – EXPRESSÕES REGULARES

As Expressões Regulares [96], [97], [98], são padrões de procura definidos para caracteres e cadeias de caracteres (*strings*). Expressões regulares são escritas numa linguagem formal que pode ser interpretada por um processador de expressão regular, que é um programa de análise sintáctica ou que examina o texto e identifica as partes que casam com a especificação dada.

Com a ajuda destes padrões pode-se pesquisar e processar o conteúdo das variáveis. Para tarefas de *Common Gateway Interface* (CGI), as expressões regulares são de importância vital. As componentes de fluxo de dados de um evento que são enviadas para o motor de correlação do SCEN, através da sua ferramenta SEC, podem ser identificadas através do uso de expressões regulares.

As expressões regulares constituem um meio muito poderoso de procura de expressões complexas em grandes volumes de dados. Da mesma forma, com ajuda das expressões regulares, é possível fazer uma correspondência (*matching*) da expressão pretendida. Esta forma é utilizada pela *pattern* do SEC.

5.7. APLICAÇÃO DE REGRAS DE CORRELAÇÃO CAUSAL AOS EVENTOS

Os eventos de entrada que serão verificados no SCEN, recorrem à utilização de expressões regulares (do tipo *RegExp*) e *strings* (do tipo *SubStr*) utilizadas na linguagem Perl. A correspondência padrão é feita a toda a sequência de dados contida no evento "*query"* e que é correspondia ou não pelo padrão de amostragem (*pattern).* Existem vários padrões que dependem do tipo de *string* de caracteres do evento de entrada. Através dos padrões pretende-se procurar correspondência aos mesmos, para inferir relações entre os eventos.

Exemplo de dois eventos diferentes de "Q*uery* Serviço":

```
Query Serviço IP: 10.0.7.44 TipoServiço: http Estado: n
Query Serviço IP: 10.0.32.71 TipoServiço: http Estado: y
```

Padrão de correspondência da regra:

```
pattern=.*IP:\s([\d\.]+)\sTipoServiço:\s(\w+)\sEstado:\sn
```

Exemplo de 2 eventos diferentes de "*Query* Servidor":

```
Query SRV IP: 10.1.33.1 SalaID: F507 Estado: y CONECTADO  Elemento: 10.1.33.100
Porto: BayStack 450-24T - 1

Query SRV IP: 10.1.33.1 SalaID: F507 Estado: y CONECTADO  Elemento: 10.0.32.100
Porto: FastEthernet0/1
```

Padrão de correspondência da regra:

```
pattern=.*SRV\sIP:\s$1\sSalaID:\s(\w+)\sEstado:\sn\s.*Elemento:\s([\d\.]+)\sPor
to:\s(.*)
```

Este padrões (*pattern)* procuram textualmente todos os caracteres, atribuindo variáveis aos valores imediatamente a seguir ao caracter ":" que vem na *string* do evento. Estes valores são as variáveis do sistema em Perl. Um esquema de trabalho contendo regras do SCEN é descrito na figura 24. Este esquema faz um conjunto de acções relativo às alíneas "a" e "b", propostas na figura 23.

Os eventos de entrada estão com setas a tracejado. Os eventos de saída que representam as acções do SCEN estão com setas a sombreado e com setas comuns. A elipse representa operação de correlação de eventos. As estrelas comuns representam os *pipes* de saída externa dos eventos e a tracejado a saída de eventos internos no sistema.

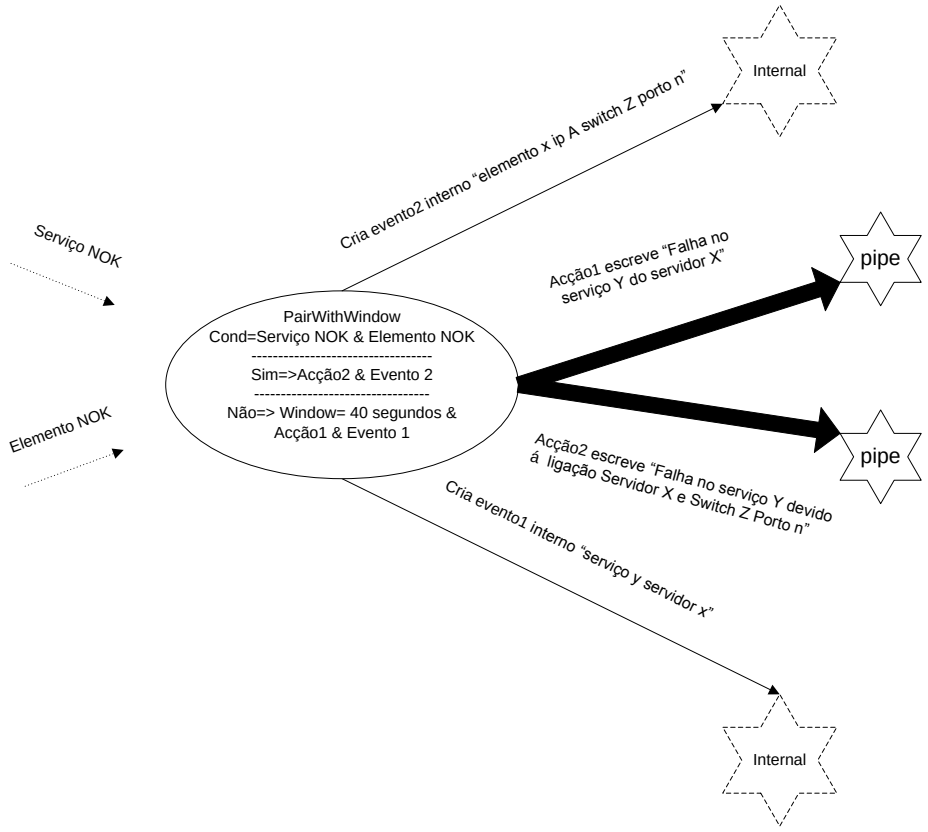


Figura 24 Algoritmo com uma regra "*PairWithWindow*" para correlação de eventos de serviço e de Elemento Rede.

O SCEN utiliza um ficheiro de configuração que contem regras e que tem como entrada um *pipe*. Para executar o ficheiro de configuração, abre-se um terminal de consola e executa-se a seguinte linha de comando, com indicação de qual o *pipe* de entrada onde o SCEN (ferramenta SEC) vai processar os dados:

```
sec -conf=ficheiro.conf -input=/directorio da pipe/pipe
```

A regra seguinte que se apresenta em modo de texto, executa o conjunto de iterações da figura 24, esta regra pertence ao ficheiro de configuração "ficheiro.conf".

```
type=PairWithWindow
ptype=RegExp
pattern=.*Servidor:\s(\d+)\sIP:\s([\d\.]+)\sTipoServiço:\s(\w+)\sEstado:\sn
desc=serviço $3 servidor $2
action=write SEC_NMSIS %t Falha no serviço $3 do servidor $2
continue2=TakeNext
ptype2=RegExp
pattern2=.*ElementoRede:\s$1\sCategoria:\s(\w+)\sIP:\s([\d\.]+)\sSalaID:\s(\w+)
\sEstado:\sn\sMAC:\s(\w+:\w+:\w+:\w+:\w+:\w+)\sNomeSwitch:\s([\d\.]+)\sPorto:\s
(\d+)
desc=elemento %1 ip $2 switch $5 porto $6
action2=write SEC NMSIS %t Falha no serviço %3 devido quebra na ligação entre
Servidor $2 e Switch $5 Porto $6
Window=40
```

O SCEN (ferramenta SEC) tem variáveis internas, essas variáveis são tiradas respectivamente da "*pattern"* e *"pattern2"* do ficheiro de configuração. Os valores assumidos na "*pattern"* podem ser reutilizados na "*pattern* 2".

Tabela 12 Descrição das variáveis internas das regras

		Ordem das variáveis		
Pattern	Origem do Valor	variável 1	variável 2	variável n
1ª pattern (pattern)	pattern pattern2	$1	$2	$n
2ª pattern (pattern2)	pattern pattern2	%1 $1	%2 $2	%n $n

Breve explicação da regra *PairWithWindow*:

(*) A regra *PairWithWindow*, contem 2 *patterns*, 2 acções e uma janela de tempo. A action2 (2º acção) é executada se e só se os eventos A (1º *pattern*) e B (2º *pattern*) ocorrerem dentro da janela de 40 segundos. Se A ocorrer e B não ocorrer, apenas a *action* (1º acção) é executada após o tempo da janela.

(*) Se a *pattern* [do evento A] é recebida seguido de outra *pattern* [evento B] dentro de 40 segundos, com o mesmo valor da variável ($1) que é o valor da primeira variável da primeira *pattern*, é escrito à saída num ficheiro de *log* (ou *pipe*), com data e hora interna do SCEN [" Falha no serviço %3 devido quebra na ligação entre Servidor $2 e Switch $5 Porto $6"], é também criado um novo evento interno ["elemento %1 ip $2 switch $5 porto $6"] e passado as outras regras.

(*) Se a 2ª *pattern* não for recebida dentro de 40 segundos é escrito num ficheiro de *log* (ou *pipe*) ["Falha no serviço $3 do servidor $2"] e criado um novo evento interno [“serviço $3 servidor $”2] que é passado às outras regras.

Para o mesmo objectivo podemos utilizar um outro tipo de regra (*Pair*), na figura 25 podemos visualizar essa regra. Este esquema tal como o outro, faz um conjunto de acções propostas na alinea (a. e b.), relativamente à figura 23. Os eventos de entrada estão com setas a tracejado. Os eventos de saída que representam as acções do SCEN estão com setas a sombreado e com setas comuns. A elipse representa operação de correlação de eventos. As estrelas comuns representam o *pipe* de saída externa dos eventos e a tracejado a saída de eventos internos no sistema.

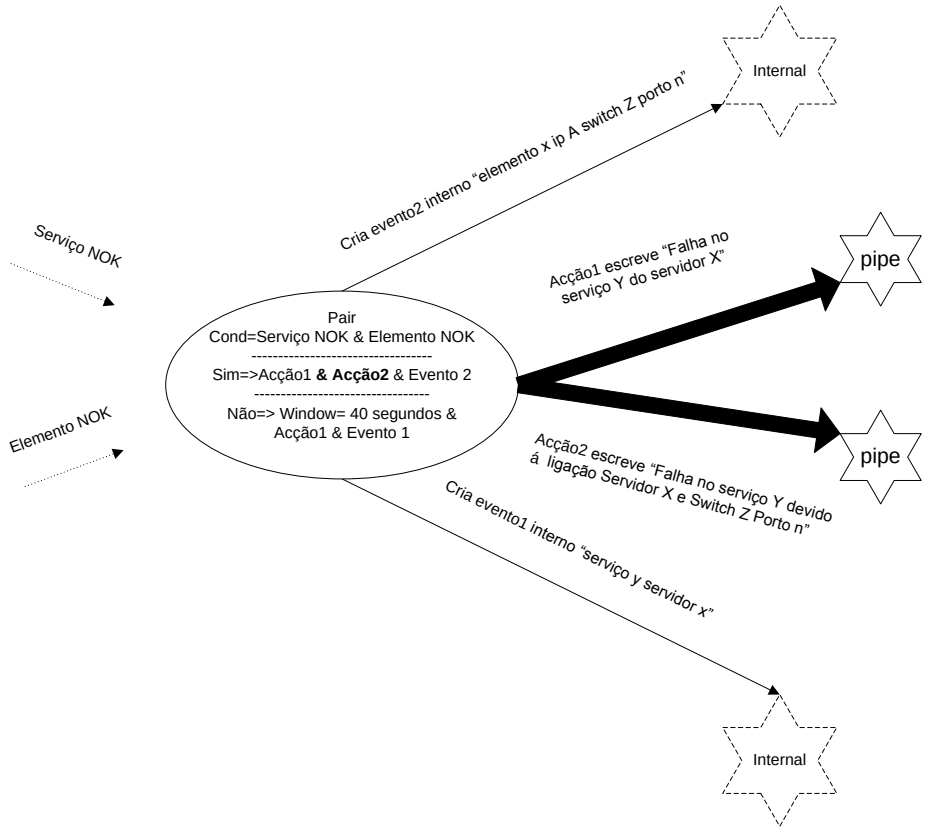


Figura 25 Algoritmo com uma regra “*Pair*” para correlação de eventos de serviço e de Elemento Rede.

Breve explicação da regra *Pair*:

(*) A regra *Pair* trata de 2 *patterns*, 2 acções e uma janela de tempo dentro da mesma regra, utiliza uma janela temporal que é activada após a primeira ocorrência do evento A. Se o evento B ocorre dentro da janela, os eventos A e B são considerados correlacionados e o *matching* é considerado à regra inteira. Se não, a operação de correlação para o par (A e B) termina. A *action* (1º acção) é executada de imediato quando é feito o *matching* do evento A e a *action2* (2ª acção) é executada quando é feito o *matching* do evento B, na mesma janela temporal.

(*) Faz *matching* à *pattern* [do evento A], se fôr recebida, escreve num ficheiro de *log* (ou *pipe*), um alarme com o seguinte texto com data e hora interna do SCEN ["Falha no serviço $3 do servidor $2"], é gerado imediatamente e um evento interno [“serviço $3 servidor $2”] e passado às outras regras.

(*) Se a 2ª *pattern* não for recebida dentro de 40 segundos, não faz mais nenhuma acção.

Quanto menor a janela temporal melhor é possível afinar a granularidade, ou seja, a quantidade de vezes que o *link* cai, dependendo desse número é possível depois mexer nos *tresholds* de contagem para definir quando se deve parar a compressão e supressão e poder inicializar um processo de generalização dos eventos.

Posto isto, é mais fácil compreender como funcionam as regras e como é possível agrupar um conjunto de regras que fazem processamento de todos os eventos recebidos à entrada, via eventos de *query* (evento Serviço, evento Servidor, evento Switch e evento Gateway). A figura 26 é um exemplo disso mesmo, é um esquema final de correlação causal de eventos, utilizando três regras em que no final é inferido apenas uma e só uma raiz da causa do problema, para um conjunto de vários tipos de eventos recebidos à entrada do sistema SCEN.

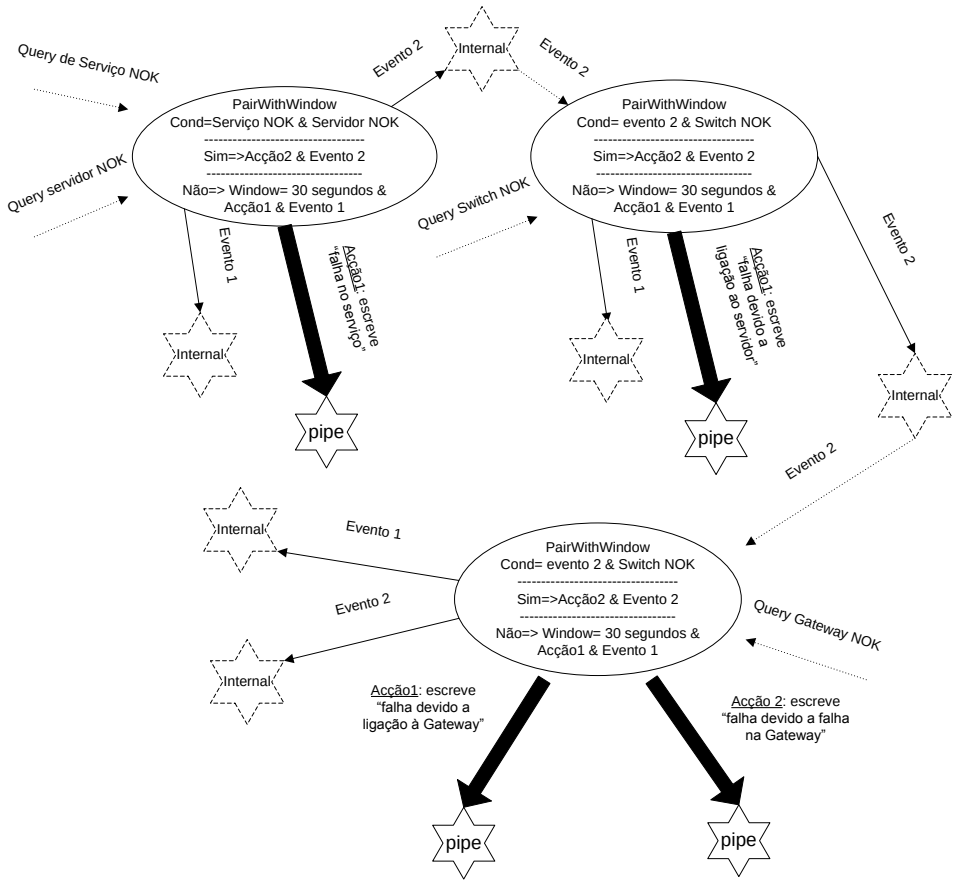


Figura 26 Algoritmo final com três regras do tipo "*PairWithWindow*" para correlação de eventos de causalidade com 4 tipos de query (Serviço, Servidor, Switch e Gateway).

5.8. APLICAÇÃO DE REGRAS DE CORRELAÇÃO TEMPORAL AOS EVENTOS

Os eventos de entrada do SCEN, via *trap* recorrem à utilização de expressões regulares. A correspondência é feita através de amostragem padrão (*pattern*) a toda a sequência de dados que aparece nos eventos de notificação (*traps*). Existem várias *patterns* dependendo do tipo de *string* de caracteres dos eventos de entrada que vão ser processados.

Um cenário que foi proposto neste trabalho é o caso típico de interfaces que estão intermitentes ao longo do tempo, ou seja, o seu estado oscila no tempo entre o estado "**down**" e estado "**up**". Para esse efeito, foi criado um conjunto de regras que fazem basicamente o seguinte:

Se em qualquer momento no tempo, fôr criado instantâneamente um evento de falha de interface de um elemento, estado passa a "**down**" e se depois, o mesmo interface do

mesmo elemento de rede passar para o estado "**up**" dentro de um espaço temporal, é criado uma relação de correlação temporal entre os dois eventos. A esta relação será inferido por correlação temporal um novo evento interno no sistema com a informação de interface "INTERMITENTE". Se o interface ficar com estado "**down**" mais do que determinado tempo e não surgir o estado "**up**", então é gerado para a saída do sistema e para o operador um alarme tipo "MAJOR DOWN", se este evento entretanto ficar com estado "**up**" então o operador receberá o alarme "NORMAL UP".

Se por outra via o evento interno "INTERMITENTE" acontecer mais do que certo número de vezes no tempo (accionado o *treshold* - Contagem*)* é inferido uma nova relação de correlação temporal interna no sistema que termina com a criação de um outro evento, de saída do sistema, a informar o operador que o interface em causa está a provocar " MAJOR Ligação Instável", ou seja, surgirem vários pares de eventos (*down/up)* que formam vários eventos internos "INTERMITENTE". Por fim, se durante determinado período de tempo o evento interno "INTERMITENTE" não acontecer mais nenhuma vez, será enviado para a saída do sistema a dar conhecimento ao operador que o interface em causa está "NORMAL Estável". Este exemplo está descrito de forma mais detalhada no modelo da figura 27, com um algoritmo responsável por executar este conjunto de tarefas, definido por um conjunto de regras.

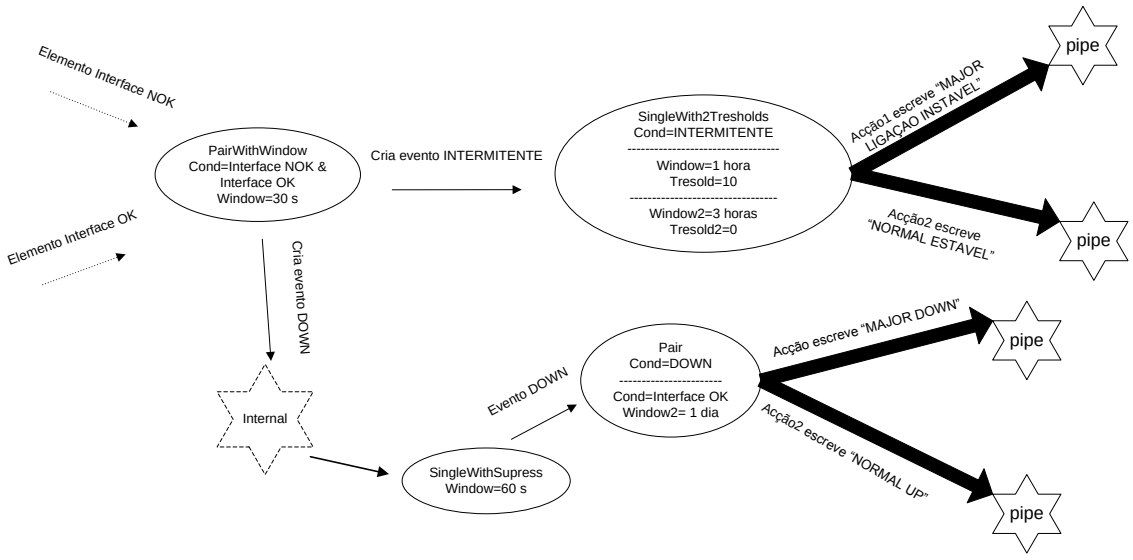


Figura 27 Algoritmo com quatro regras "*PairWithWindow", "SingleWithSupress", "Pair" e "SingleWith2Treshold"* para correlação temporal de eventos via notificação (trap).

Este cenário é bem evidente das capacidades do sistema proposto neste trabalho, com correlação de alarmes em tempo real de acordo com a sua janela temporal e de forma a suprimir grande quantidade de eventos. Paralelamente realiza operações de contagem a determinados eventos e define *tresholds* máximos de repetição.

5.9. RESPOSTA DO SISTEMA À SAÍDA

O sistema de saída do SCEN está projectado para fazer filtragem de eventos, compressão de eventos, supressão de eventos, fazer contagem de eventos, utilizar *tresholds* máximos de eventos, fazer generalização, especialização e ter uma componente de relação temporal dos eventos que chegam à entrada.

O objectivo deste trabalho nos sistemas de regras de correlação causal é dar uma resposta mais suscinta com informação mais verdadeira sobre a raiz da causa das falhas. O objectivo dos sistemas de regras temporais é fazer uso dos benefícios de uma aproximação temporal. Por exemplo, fazer filtragem, contagem e compressão aos eventos, materializando a resposta do sistema com características mais genéricas do que se está a passar a cada momento e tratar de um número muito elevado de eventos específicos.

O SCEN analisa os eventos que entram num ficheiro de entrada normalizado, sejam esses eventos via *query* às bases dados do NMSIS através do *daemon* PHP, sejam os eventos recebidos directamente da rede via *trap* através do *daemon* "*snmptrapd"*. O motor de correlação, recebe simultâneamente os dois tipos de eventos eventos (*Query* e *trap*). No caso das bases de dados o canal de comunicação da informação é bidirecconal, no caso das *traps* o canal é unidirecional. O sistema processa os eventos que estão a decorrer na rede e responde com informação coerente correcta e verdadeira ao utilizador sobre o que está acontercer a cada instante.

O operador é informado do estado da rede, através de uma página WEB com a listagem de todos os eventos que estão a decorrer em tempo real e também através de notificação via mail, caso se justifique.

5.9.1. NOTIFICAÇÃO VIA WEB

O SCEN permite monitorizar em tempo real todas as mensagens de saída do sistema, tanto num ficheiro de texto como numa página WEB. Após o SCEN executar uma acção de escrita num ficheiro de saída, que pode ser do tipo *file stream* ou de *log*. É possível retirar

essas mensagens de saída e redirecioná-las para uma página *HyperText Markup Language* (HTML) mais amiga do utilizador. Essa página permite colectar todas as mensagens de saída do sistema.

Foi instalado um servidor *HyperText Transfer Protocol* (HTTP), chamado "*apache2*". No directorio raiz do *apache2* foi criado uma página com um script em PHP, que abre o ficheiro de saída do sistema, lê todo o texto linha-a-linha e visualiza-o em formato HTML.

A página faz um refresh automático a cada 5 segundos, como ilustra a figura 28.

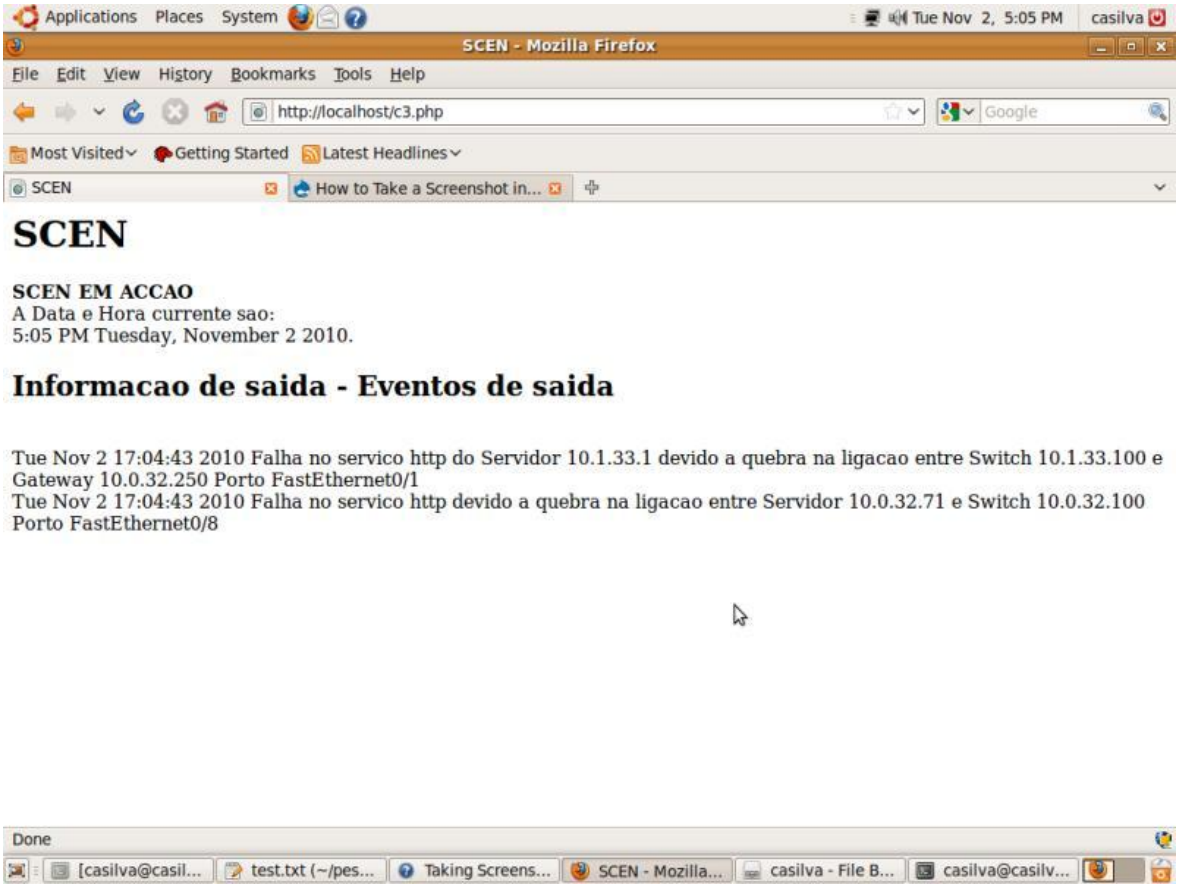


Figura 28 Págima WEB de eventos de saída do sistema SCEN.

5.9.2. NOTIFICAÇÃO VIA MAIL

O sistema de saída está preparado para enviar mensagens de mail. Através das acções do SCEN pode-se escrever as mensagens de saída numa *file stream ou* num ficheiro de *log* e é

possível até executar comandos de shell. Para enviar uma mensagem de mail, a acção "*write*" de escrita num ficheiro é substituída pela acção de "*pipe*". O "pipe" dos sistemas Unix, é um comando de *shell* que executa um programa ou um *script* e o resultado é realimentado à sua entrada.

Foi preciso instalar um *Mail User Agent* (MUA) para o mail local interno da máquina do SCEN e um *Mail Transport Agent* (MTA) para possível ligação a outro MTA do *Internet Service Provider* (ISP), através do protocolo Simple *Mail Tranfer Protocol* (SMTP). O MUA utilizado e instalado foi o"*mailx*" e o MTA utilizado foi o "*postfix*". O "*mailx*" é apenas um MTU local, este agente irá deixar todas as mensagens de saída de mail na fila de espera utilizado pelo MTA do sistema (neste caso o *postfix* que vêm por defeito com o *ubunto*). O "*postfix*" é o agente que tem a tarefa mais pesada de enviar o mail aos outros MTAs.

Para configurar o postfix para usar o servidor de SMPT do ISP, é preciso alterar a configuração do "*postfix*" no directório */etc/postfix/main.cf* e definir a seguinte configuração, depois fazer *reload* ao executável "*postfix*" (*sudo /etc/init.d/postfix restart*):

```
relayhost = isp.smtp.server.domain.name
```

Posto isto, é possível enviar mensagens de mail ao utilizador local e/ou utilizadores remotos. Pode-se enviar uma mensagem do tipo:

```
echo 'corpo da mensagem' | mailx [-s subject] mail-destino
…
echo 'Thu Sep 16 17:28:37 2010 Falha no serviço http do servidor 10.1.33.1' |
mailx –s "Evento SCEN_NMSIS" root@localhost
```

Com o sistema de regras do SCEN é possível através da acção *pipe*, fazer exactamente a mesma operação do comando Unix anterior, da seguinte forma:

```
action=pipe'%t Falha no serviço $2 do servidor $1' | mailx –s "Evento
SCEN NMSIS" root@localhost
```

5.10. SIMULAÇÃO DE TESTES E RESULTADOS

5.10.1. DEFINIÇÃO DO TIPO DE TESTES

Os testes consistiram em validar a consistência do sistema montado, na percepção e relevância que dá a determinada sequência de eventos, que levam à falha de um serviço. É analisado a forma como o sistema se comporta face a essas entradas e se age em conformidade.

O objectivo principal nesta série de testes e simulações é a descoberta da raiz da causa dos problemas observados. Essa descoberta deve ser correcta e verdadeira, ou seja, o mais aproximado da causa real das falhas subjacentes. Para o efeito, foi criado um serviço HTTP a correr numa máquina "10.1.33.1", identificada na figura 29. Este serviço será monitorizado ao mesmo tempo que será condicionado através da execução de um conjunto de acções que visam a interrupção do serviço, através de cortes fisicos em vários segmentos de rede. O algoritmo composto por regras tem a função de notificar o operador de sistema com o mínimo de informação, o mais verdadeiro e credível quanto possível.

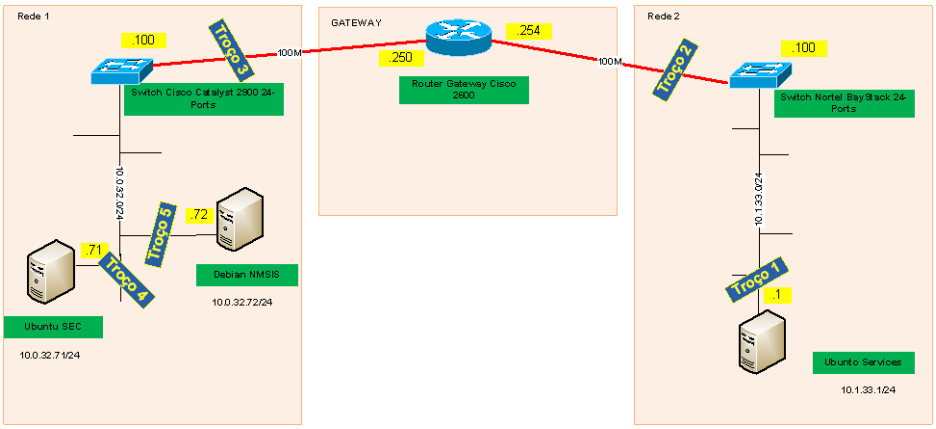


Figura 29 Planta de testes.

5.10.2. TESTE - 1

Se fôr desactivado o serviço na máquina 10.1.33.1. O sistema analisa à sua entrada os quatro tipos de eventos via *query*. Correlaciona todos os eventos e responde com um novo evento de saída, de forma a ser visualizado na página WEB ou por notificação de *mail* para o operador, a informar que o serviço falhou.

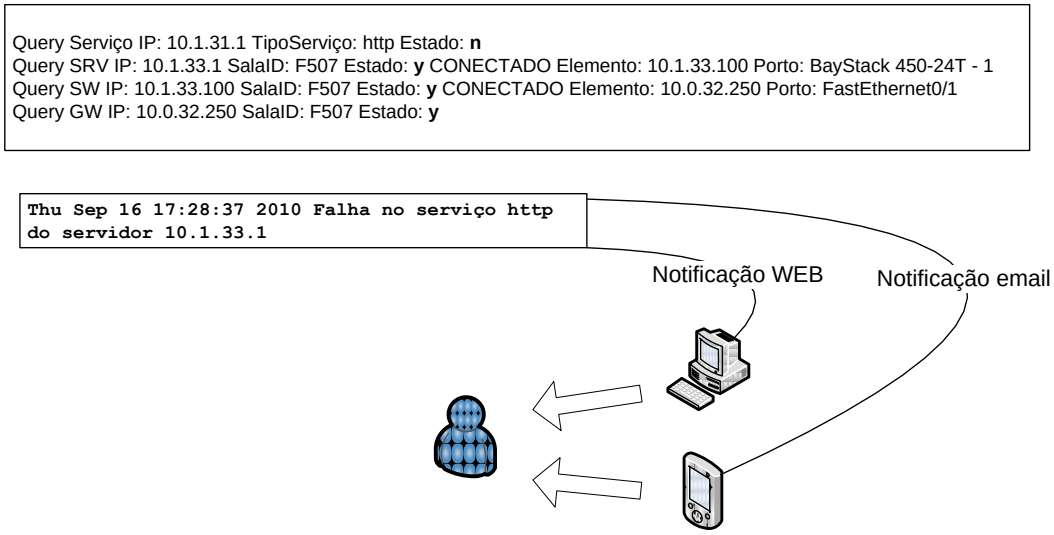


Figura 30 Simulação da resposta do sistema ao Teste 1

5.10.3. TESTE – 2

Se fôr cortado o segmento de rede (troço 1) entre a máquina 10.1.33.1 e o porto "1" do *Switch* 10.1.33.100. O sistema analisa os quatro tipos de eventos via *query* e responde com um novo evento de saída, que pode ser visualizado na página WEB ou por notificação de *mail* para o operador, a informar que o serviço falhou, devido a corte na ligação entre a máquina que tem o serviço e o respectivo porto do *switch* onde está ligado.

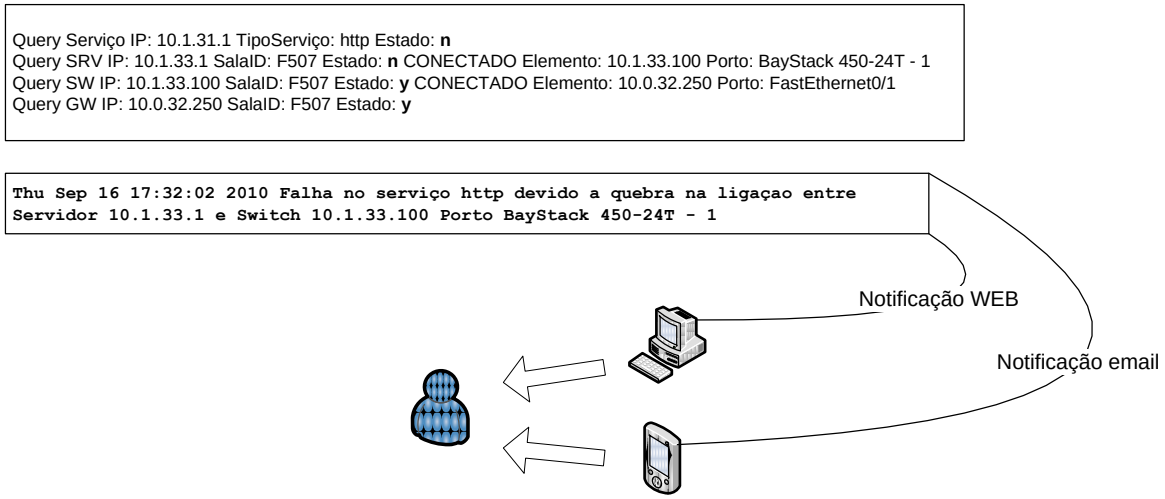


Figura 31 Simulação da resposta do sistema ao Teste 2

5.10.4. TESTE – 3

Se fôr cortado o segmento de rede (troço 2) entre o *switch* 10.1.33.100 e o porto F0/1 da *gateway* 10.0.32.250. O sistema analisa os quatro tipos de eventos via *query* e responde com um novo evento de saída, que pode ser visualizado na página WEB ou por notificação de *mail* para o operador, a informar que o serviço falhou, devido a corte na ligação entre o *switch* e o respectivo porto da gateway onde está ligado.

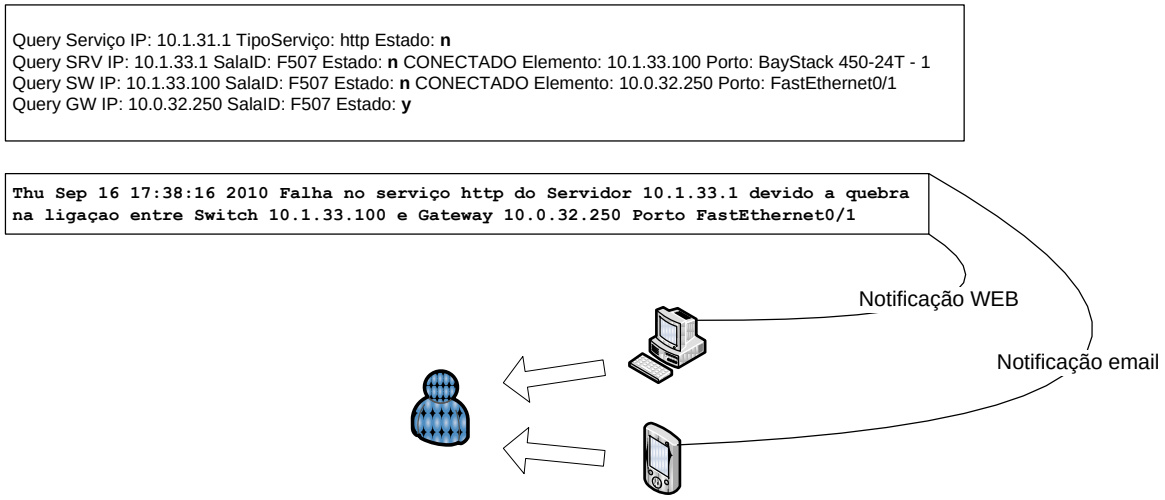


Figura 32 Simulação da resposta do sistema ao Teste 3

5.10.5. TESTE – 4

Se fôr cortado o segmento de rede (troço 3) entre o *switch* 10.0.32.100 e porto F0/0 da *gateway* 10.0.32.250. O sistema analisa os quatro tipos de eventos via *query* e responde com um novo evento de saída, que pode ser visualizado na página WEB ou por notificação de *mail* para o operador, a informar que o serviço falhou, devido a falta de acessibilidade ao *gateway* (*Router* G*ateway* isolado).

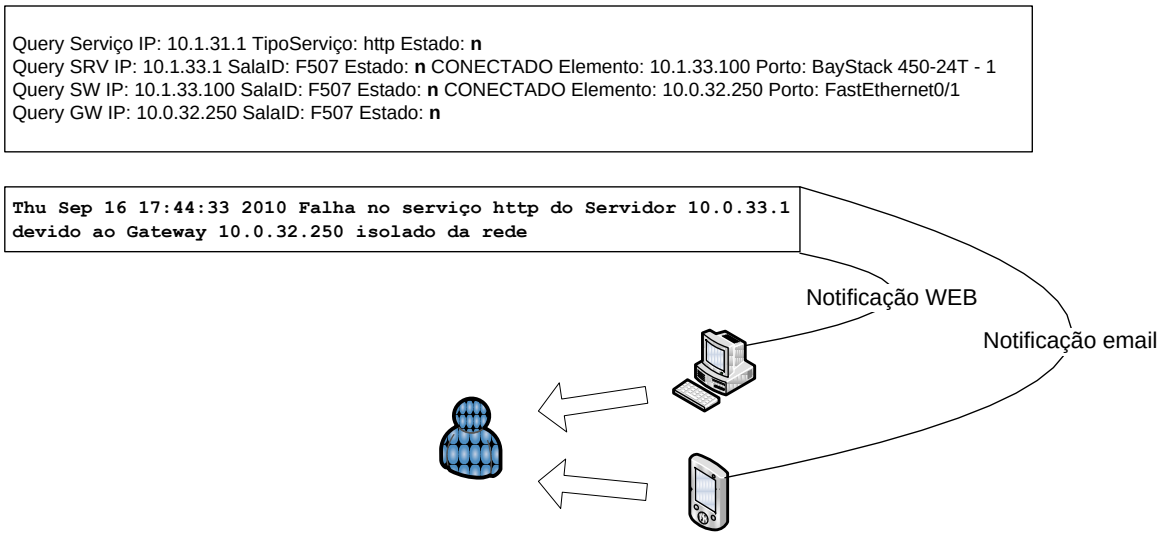


Figura 33 Simulação da resposta do sistema ao Teste 4

5.11. PERFORMANCE DA INFRA-ESTRUTURA

Um dos estudos de grande importância que se pode submeter a um novo sistema, são os que fazem análise da performance do sistema, ou seja, a resposta do sistema aos vários tipos de entradas.

Vários aspectos podem influenciar a performance do sistema SCEN:

- ✓ Tipo de plataforma de hardware;
- ✓ Tamanho do conjunto de regras;
- ✓ Complexidade das expressões regulares em Perl;

- ✓ Tipo de acções do SEC;
- ✓ A Velocidade dos dados na entrada do sistema SCEN;
- ✓ Carga de processamento no CPU;
- ✓ Tipo de sistema de armazenamento *Backend* (base de dados ou equivalente);
- ✓ Congestionamento da Rede.

Todos estes aspectos limitam a capacidade de resposta do sistema SCEN. Podia ter sido feito o estudo ao *throughput* do sistema SCEN, através de medidas de: percentagem; detecção das falhas; veracidade das causas das falhas inferidas; capacidade de compressão e de supressão das falhas.

Como este trabalho é feito em ambiente de teste e como apenas se pretende ver o sucesso dos obectivos atingidos, ou seja, se o sistema é capaz de inferir relações entre os eventos, para chegar a raíz da causa das falhas de rede, estabelecendo correlação causal e temporal entre os eventos que processa à sua entrada.

Nesta linha de pensamento, a medição de *througput* do sistema SCEN pareceu uma questão secundária. No entanto, o autor [74], [123], com uma configuração com 63 regras: 29 Single, 9 Pair, 1 PairWithWindow, 5 SingleWithThreshold, 17 SingleWith2Threshold, and 2 Suppress, conseguiu grandes resultados de performance, com apenas um computador 200MHz Intel Pentium processor5, com 32Mb de memória, e sistema operativo Linux.

6. CONCLUSÕES

6.1. CONCLUSÕES GERAIS

O objectivo do trabalho realizado foi a definição e criação de uma infra-estrutura de monitorização com capacidade para correlacionar (filtrar, comprimir, suprimir, generalizar, especializar, etc), de forma autónoma e inteligente os eventos recebidos a partir de sistemas informáticos. Existem muitas soluções, umas de software aberto, comercial e outras com soluções em ambientes fechados, muitas das quais bastante elaboradas e complexas.

O assunto relacionado com gestão de eventos, onde se insere a temática desta Tese de dissertação, tem já cerca de duas décadas de estudo, inúmeros desenvolvimentos e trabalho realizado por vários autores e organizações, que têm feito esforço por debater e solucionar muitas das questões que têm surgido ao longo do tempo. Existem inúmeros modelos e projectos neste âmbito de análise, nenhum só por si conseguiu resolver totalmente todas as questões relacionadas com correlação de eventos em gestão de redes e sistemas. Muito disto, deve-se ao facto de os sistemas e as redes estarem em constante transformação, a evoluírem a fundirem-se continuamente, o que torna a sua caracterização e definição, práticamente impraticáveis tecnicamente.

A infra-estrutura de correlação de eventos proposta, o sistema SCEN, visa assim preencher de uma lacuna que existe em muitas redes e que é um tema muito actual e discutido. O exercício proposto neste trabalho foi, de alguma forma, criar um módulo com valor acrescentado a um sistema *legacy* já instalado, aceitando esse facto e não o limitar nem desviar em nada essa plataforma *legacy* das suas principais funções. Para validar o modelo foi implementado um laboratório de teste virtual onde foram replicados os elementos constituintes de uma rede real e onde foi introduzido a infra-estrutura proposta. Foi executado uma bateria de testes e simulações para validação de resultados.

Como conclusão, a infra-estrutura que foi definida e testada reflecte o funcionamento pretendido. Teve por base o estudo de gestão de redes com protocolo SNMP e o estudo de várias ferramentas com funções de correlação de eventos. Baseando-se apenas nas definições pré-existentes sobre os serviços monitorizados e o conhecimento de uma ferramenta específica (NMSIS), foi criado um novo módulo (SCEN) que acrescenta um maior leque de opções nesta matéria de estudo ao sistema anterior (*legacy)*.

Este novo módulo com grande versatibilidade e escalabilidade, pode em pleno funcionamento, aprender com novas alterações do sistema e com a introdução de novos eventos. De forma contínua, pode decidir a relevância das falhas de rede e assim aplicar uma resposta correcta e eficaz em tempo real. Existem várias formas de avaliar e analisar os problemas da disponibilidade e performance de uma infra-estrutura e que esta forma poderá ser também um complemento para uma infra-estrutura mais global ou poderá ser por si só uma base conceptual para uma infra-estrutura auto-regulável e auto-adaptável mais avançada.

6.2. DIFICULDADES E DESAFIOS

Existiram várias dificuldades que surgiram na elaboração desta Tese de dissertação, uma das quais não posso deixar de referir foi efectivamente a falta de tempo sentida. Quer pelo carácter não profissional que a elaboração desta dissertação constituiu, dado que foi elaborada, dentro dos períodos laborais, nos períodos pós-laborais e de férias. Também a falta de tempo, característica própria dos tempos actuais, que comprende a actividade profissional a família, os filhos, etc.

Sob o ponto de vista de pesquisa, não há práticamente informação nenhuma, sobre esta temática em língua portuguesa, as que encontrei quase sempre foram com informação

muito específica sobre os sistemas monitorizados. No que toca ao módulo adoptado, embora existissem outras aprocximações, considerei este modelo o mais genérico e completo possível e com uma resposta muito capaz e eficaz.

Penso ter-se reunido as condições necessárias para responder às questões levantadas pelos meus orientadores e que serviram de alicerces de base conceptual deste trabalho. Pergunta essas, que deixo aqui referenciadas: "*Como correlacinar uma grande quantidade de eventos que podem surgir no sistema NMSIS quando na verdade, apenas, uma falha pode ser a causa do problema*". A solução foi a integração de uma ferramenta com capacidade de correlação, que responde de forma eficaz à questão levantada.

Ficou por fazer o desenvolvimento e adaptação da resposta do SCEN no ambiente gráfico do sistema *legacy* (NMSIS) para fazer a interface com o operador. No entanto, foi atingido o objectivo do operador receber os eventos de saída do SCEN de uma forma visualmente amigável e limpa. Esse objectivo foi concretizado através da visualização de uma página WEB do SCEN e via mensagens de *mail* para o operador do sistema.

6.3. CONSIDERAÇÕES FINAIS

No panorâma actual, as principais dificuldades que os operadores de plataformas de gestão de alarmes se prendem. É a grande quantidade de informação que muitas vezes tem ao seu dispor. Não exactamente a informação técnica que explica como proceder num determinado contexto, mas na informação que chega dos sistemas que tem que gerir. Pretende-se que a quantidade de eventos seja menor e com informação mais útil sobre o que está realmente acontecer em tempo real. Não só porque hoje os equipamentos têm mais capacidade de serviços e as aplicações trocam mais mensagens e geram mais registos, mas porque cada vez mais as aplicações dependem mais umas das outras.

Com a abundância de informação vaga e incompleta, as mensagens sobre os sistemas de gestão é elevada. Quando se interroga alguém ligado à gestão de redes, como é que é feita a gestão dos sistemas e recursos a seu cargo, muitas vezes as respostas podem ser preocupantes. A maior parte das empresas não gerem de forma eficiente os seus recursos e muitas vezes não sabe efectivamente como o fazer, por isso não pode tirar dividendos de uma gestão eficaz.

É fácil apregoar que é indispensável ter um sistema de gestão, mas é um grande desafio justificar os custos e as tarefas necessárias para uma correcta gestão dpos sistemas. Para o fazer é obrigatório ter equipas dedicados a essas tarefas e em particular em empresas de menor dimensão pode ser uma limitação. Ultrapassar o problema da recolha e análise eficaz dos eventos gerados pelos sistemas e redes informáticas no domínio da computação, é por si só, uma questão complexa que requer a implementação de um sistema simples e eficaz de identificação e resolução de problemas.

Como é apanágio da Engenharia "grandes problemas requerem soluções simples". Existem inúmeros trabalhos de investigação que podem ser encontrados, onde se abordam estas questões no sentido de criar e melhorar os processos que levem a uma eficaz gestão de recursos e de sistemas. Esta dissertação é mais um desses trabalhos e mais um exercício nesse sentido.

6.4. TRABALHO FUTURO

Partindo do princípio que "Nenhuma obra nasce acabada", existem sempre melhoramentos a fazer, alguns deles podem ser discutidos neste capítulo.

6.4.1. DESENVOLVIMENTO DE UM AMBIENTE GRÁFICO

Na óptica da interface gráfica do sistema e com o propósito de criar um ambiente *user friendly* com o operador do sistema *legacy* (NMSIS), faltou integrar a saída do sistema SCEN na interface gráfica nativa do NMSIS. Embora a resposta do sistema SCEN seja visualizado numa página WEB concebida para o efeito, ficou por fazer a interface de saída do sistema de correlação SCEN no ambiente NMSIS.

6.4.2. SISTEMA DISTRIBUIDO

Dotar o sistema com capacidade de distribuição, com vários processos activos a correr em simultâneo. Ou seja, vários ficheiros de configuração a trabalhar autonomamente podendo estabelecer ligações através de eventos externos ou contextos. O sistema pode interagir com outros sistemas e scripts, activando várias acções com comandos de *shell* que podem ser utilizados nas mais diversas circunstâncias.

6.4.3. CRESCIMENTE E ESCALONAGEM DO SISTEMA

É possível fazer o sistema crescer em sistemas e subsistemas, não existe qualquer limitação nesse sentido. Podem ser adicionadas continuamente mais regras de correlação à sua configuração.

6.4.4. OUTRAS ÁREAS DE ACÇÃO

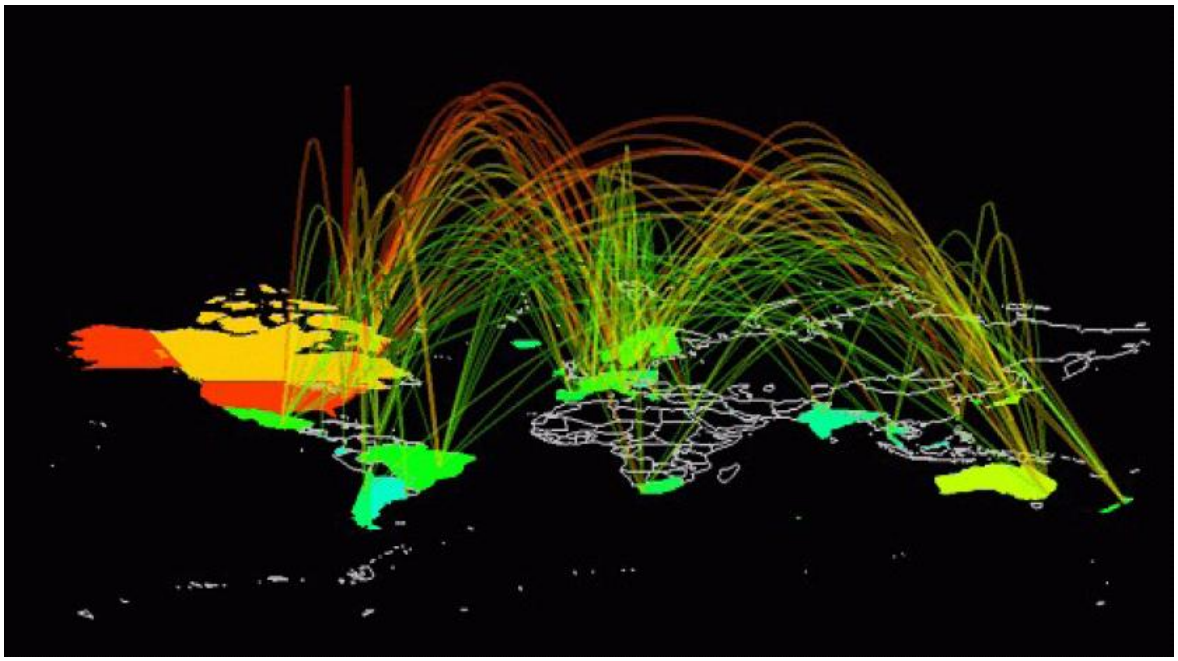
Pode-se alargar o campo de análise e acção a todas as camadas e respectivos protocolos. Por exemplo, protocolos de comunicação e de encaminhamentos da informação entre os elementos constitintes da rede ou análise de eventos de *log* da própria máquina.

Lê, e facilmente esqueces…

Assiste, e talvez te lembres…

Faz, e jamais te esquecerás.

(Provérbio de origem indiana)



Referências Documentais

[1] J. Case, R. Mundy, D. Partain, and B. Stewart. Introduction to Version 3 of the Internet-standard Network Management Framework. RFC 2570, SNMP Research, TIS Labs at Network Associates, Ericsson, Cisco Systems, April 1999.

[2] J. Case, M. Fedor, M. Schoffstall, and J. Davin. A Simple Network Management Protocol. RFC 1157, SNMP Research, PSI, MIT, May 1990.

[3] M. Rose and K. McCloghrie. Structure and Identification of Management Information for TCP/IP-based Internets. RFC 1155, Performance Systems International, Hughes LAN Systems, May 1990.

[4] M. Rose and K. McCloghrie. Concise MIB Definitions. RFC 1212, Performance Systems International, Hughes LAN Systems, March 1991.

[5] M. Rose. A Convention for Defining Traps for use with the SNMP. RFC 1215, Performance Systems International, March 1991.

[6] K. McCloghrie, D. Perkins, J. Sch¨onw¨alder, J. Case, M. Rose, and S. Waldbusser. Structure of Management Information Version 2 (SMIv2). RFC 2578, Cisco Systems, SNMPinfo, TU Braunschweig, SNMP Research, First Virtual Holdings, International Network Services, April 1999.

[7] K. McCloghrie, D. Perkins, J. Sch¨onw¨alder, J. Case, M. Rose, and S. Waldbusser.Textual Conventions for SMIv2. RFC 2579, Cisco Systems, SNMPinfo, TU Braunschweig, SNMP Research, First Virtual Holdings, International Network Services, April 1999.

[8] K. McCloghrie, D. Perkins, J. Sch¨onw¨alder, J. Case, M. Rose, and S. Waldbusser. Conformance Statements for SMIv2. RFC 2580, Cisco Systems, SNMPinfo,TU Braunschweig, SNMP Research, First Virtual Holdings, SNMP Research, International Network Services, April 1999.

[9] D. Harrington, R. Presuhn, and B. Wijnen. An Architecture for Describing SNMP Management Frameworks. RFC 2571, Cabletron Systems, BMC Software, IBM T. J. Watson Research, April 1999.

[10] K. Chan, J. Seligson, D. Durham, S. Gai, K. McCloghrie, S. Herzog, F. Reichmeyer, R. Yavatkar, and A. Smith. COPS Usage for Policy Provisioning (COPS-PR). RFC 3084, Nortel Networks, Intel, Cisco, IPHighway, PFN, Allegro Networks, March 2001.

[11] K. McCloghrie, M. Fine, J. Seligson, K. Chan, S. Hahn, R. Sahita, A. Smith, and F. Reichmeyer. Structure of Policy Provisioning Information (SPPI). RFC 3159, Cisco Systems, Nortel Networks, Intel, Allegro Networks, PFN, August 2001.

[12] J. Sch¨onw¨alder and F. Strauß. Next Generation Structure of Management Information for the Internet. In Proc. 10th IFIP/IEEEWorkshop on Distributed

Systems: Operations and Management, pages 93–106. Springer Verlag, October 1999.

[13] J. Sch¨onw¨alder and A. M¨uller. Reverse Engineering Internet MIBs. In Proc.7th IFIP/IEEE International Symposium on Integrated Network Management, Seattle, May 2001.

[14] W. Yeong. SNMP Query Language. Technical Report 90-03-31-1, Performance Systems International, March 1990.

[15] A. V. Aho, B. W. Kernighan, and P. J. Weinberger. The AWK Programming Language. Addison Wesley, 1988.

[16] J. K. Ousterhout. Tcl and the Tk Toolkit. Addison-Wesley, April 1994.

[17] J. Sch¨onw¨alder and H. Langend¨orfer. Tcl Extensions for Network Management Applications. In Proc. 3rd Tcl/Tk Workshop, pages 279–288, Toronto, July 1995.

[18] J. Sch¨onw¨alder. Married with Tcl. In Proc. 1st European Tcl/Tk User Meeting, Hamburg, June 2000.

[19] M. T. Rose and K. McCloghrie. How to Manage Your Network Using SNMP. Prentice Hall, 1995.

[20] L. Wall, T. Christiansen, and J. Orwant. Programming Perl. O'Reilly, 3 edition, July 2000.

[21] J Sch¨onw¨alder. Specific Simple Network Management Tools. In Proc. LISA 2001, December 2001.

[22] T. Oetiker. "MRTG - Multi Router Traffic Grapher". In Proc. 12th Conference on Large Installation System Administration (LISA XII), Boston, December 1998.

[23] J. Sch¨onw¨alder, "Evolution of Open Source SNMP Tools". University of Osnabr¨uck, Germany. In Proc. LISA 2002, April 29.

[24] A. Chourmouziadis and G.Pavlou. "Web Services Monitoring: An Initial Case Study on the Tools". In Network Operations and Management Symposium, 2008. NOMS 2008. IEEE7-11 April 2008 Page(s):827 - 830.

[25] [J. Clark, S. DeRose, "XML Path Language v 1.0", W3C recommendation, November 1999.

[26] A. Berglund, S. Boag et al, "XML Path Language v 2.0", W3C recommendation, January 2007.

[27] A. Chourmouziadis et al "Efficient Information Retrieval in Network Management Using Web Services" DSOM 2006 Dublin, Ireland.

[28] H. Asgari et ak "Scalable monitoring support for resource management and service assurance", IEEE Network, Vol 18, Iss. 6. 2004 pp.6 – 18.

[29] CADENUS, http://wwwcadenus.fokus.fraunhofer.de/

[30] ENTHRONE, http://www.enthrone.org/, 2nd phase 1/9/2006.

[31] A. Pras, T. Drevers, R. van de Meent, D. Quartel. Comparing the Performance of SNMP and Web Services-Based Management, Network and Service Management, IEEE Transactions on Volume 1, Issue 2, Dec. 2004 Page(s):72 – 82.

[32] J. Sch¨onw¨alder, “Overview of the 2002 IAB Network Management Workshop,” May 2003, IETF RFC 3535.

[33] J. Sch¨onw¨alder, A. Pras, and J.P. Martin-Flatin, “On the future of Internet management technologies,” IEEE Communications Magazine, vol. 41, no. 10, pp. 90–97, October 2003.

[34] “Homepage of the XML Protocol WG”, http://www.w3.org/2000/xp/Group/

[35] “Homepage of the Web Services Description WG,” http://www.w3.org/2002/ws/desc/

[36] “Homepage of the IRTF Network Management Research Group (NMRG),” http://www.ibr.cs.tu-bs.de/projects/

[37] J. van Sloten, A. Pras, and M. van Sinderen, “On the standardisation of Web service management operations,” in Proceedings of 10th Open European Summer School School and IFIP WG 6.3 Workshop (EUNICE 2004), June 2004, pp. 143–150.

[38] Paula Viana, “Modelo TCP/IP e Modelo TCP/IP Encapsulamento”, INGRE, IPv4-v2.2, 2005.

[39] Cisco Systems, “Understanding TCP/IP”, Appendix A, Posted: Sat Sep 28 PDT 2002.

[40] Paulo Pereira, Jorge Sousa, Pedro Teixeira, Paulo Pinto, “Plataforma Dinâmica de Gestão de redes”, Instituto Superior Técnico INESC, Faculdade de Ciências e Tecnologia Universidade Nova de Lisboa, publicado nas Actas da 1ª Conferência Nacional de Redes de Computadores CRC'98.

[41] Kraig Meyer, Mike Erlinger, Joe Betser, Carl Sunshine, Germán Goldszmidt, Yechiam Yemini. Decentralizing Control and Intelligence in Network Management. Proceedings 4th International Symposium on Integrated Network Management, Santa Barbara, CA, Maio de 1995.

[42] Jürgen Schönwälder. Using Multicast-SNMP to Coordinate Distributed Management Agents. 2nd IEEE Workshop on Systems Management, Toronto, 20th de June 1996.

[43] João Pires, “TMN- Telecommunication Managment Network”. Redes de gestão de Telecomunicações, Redes de Telecomunicações, 2009, 2010.

[44] Martin P. Clark , “ Networks and Telecomunications”, cap. 27,John Wiley & Sons, 1997

[45] Salah Aidarous and Thomas Plewyak, “ Telecomunications Network Management in to 21st Century,” IEEE Press, 1993

[46] R. E. Caruso,“ Network Management: A Tutorial Overview,” IEEE Communications Magazine, Março 1990, pp. 20-25.

[47] Dan Plakosh, Network Management: An Overview, Software Engineering Institute,September 2, 2000, can be found from: http://www.sei.cmu.edu/str/descriptions/network_body.html

[48] ITU-T, M.3010 Recommendation, Principals for a Telecommunications Management Networks, 02/2000.

[49] Mousa Al-Mohammed, "Distributed Object Technologies for Element Management System", September 10, 2003

[50] J. Case, K. McCloghrie, M. Rose, S. Waldbusser, “Manager Information Base for version 2 of the Simple Network Management Protocol”, RFC 1450, 1993

[51] GRANVILLE, L. Z.; *Construção de agentes SNMP em ambiente Linux*. Instituto de Informática, Universidade Federal do Rio Grande do Sul. Agosto, 2006.

[52] Net-SNMP, Version: 5.2.1, http://www.ne.snmp.org/

[53] SEC - simple event correlator, http://simple-evcorr.sourceforge.net/

[54] NAGIOS, http://www.nagios.org

[55] BARTH Wolfgang, System and Network Monitoring. NAGIOS, Copyright 2006 Open Source Press GmbH.

[56] OpenNMS, http://www.opennms.org

[57] Product tour: http://www.sortova.com/tools/opennms

[58] OpenNMS, FAQ: http://faq.opennms.org/faq/fom-serve/cache/1.html

[59] Documentações: http://www.opennms.org/users/docs/

[60] Listas de discussão: http://www.opennms.org/users/discuss/

[61] RRDTOOL: http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/

[62] Cacti: http://www.raxnet.net/products/cacti/

[63] Cacti. Alexandro Silva. 2008, http://openmaniak.com/pt/cacti.php

[64] Cacti Documentation site, http://www.cacti.net

[65] Costa, Felipe, Ambiente de rede monitorado com Nagios e Cacti, 2008.

[66] Jhony Maiki Maseto, Soluções de Sofware Livre para Monitoração de Redes, 2008.

[67] MULTI ROUTER TRAFFIC GRAPHER On-line: URL http://www.mrtg.org

[68] MULTI ROUTER TRAFFIC GRAPHER. On-line: URL http://oss.oetiker.ch/mrtg

[69] OTSUKA Joice Lee. MIB – management Information base.

[70] TANENBAUM, Andrew. Redes de Computadores. Primeira Edição.Editora Campus, 1994.

[71] MCCLOGHRIE, K.; ROSE, M.; Management Information Base for Network Management of TCP/IP-based internets. Request For Comments 1066 – Internet Engineering Task Force, (August, 1988).

[72] MCCLOGHRIE, K.; ROSE, M.; Management Information Base for Network Management of TCP/IP-based internets: MIB-II. Request For Comments 1213 – Internet Engineering Task Force, (March, 1991).

[73] Risto Vaarandi; Platform Independent Event Correlation Tool for Network Management. Proceedings of the 2002 IEEE/IFIP Network Operations and Management Symposium, pp.

[74] Risto Vaarandi; SEC - a Lightweight Event Correlation Tool. Proceedings of the 2002 IEEE Workshop on IP Operations and Management, pp. 111-115, 2002

[75] G. Jakobson, M. Weissman, L. Brenner, C. Lafond, C. Matheus, “GRACE: Building Next Generation Event Correlation Services”,Proceedings of the 7th Network perations and Management Symposium, pp. 701-714, April 2000.

[76] Stuart Staniford, James A. Hoagland, Joseph A. McAlerney, “Practical utomated Detection of Stealthy Portscans”, in press (to appear in the Journal of Computer Security), Silicon Defense, 513 2nd Street, Eureka, CA 95501, USA, 1999.

[77] S. Staniford-Chen et al., “GrIDS – A Graph-Based Intrusion Detection System for Large Networks”, Proceedings of the 19th National Information Systems Security Conference, pp. 361-370, October 1996.

[78] Martin Roesch, “Snort – Lightweight Intrusion Detection for Networks”, Proceedings of USENIX 13th System Administration Conference, pp. 229-238, November 1999.

[79] Stephen E. Hansen and E. Todd Atkins, “Automated System Monitoring and Notification With Swatch”, Proceedings of USENIX 7th System Administration Conference, pp. 145-152, November 1993.

[80] Wolfgang Ley and Uwe Ellerman, logsurfer(1) and logsurfer.conf(4) manual pages, unpublished, see http://www.cert.dfn.de/eng/logsurf/

[81] Matt Bing and Carl Erickson, “Extending UNIX System Logging with SHARP”, Proceedings of USENIX 14th System Administration Conference, pp. 101-108, December 2000.

[82] Hewlett-Packard Company, Event Correlation Services – Designer’s Guide, HP document J1095-90304, 1998.

[83] SMARTS, http://www.smarts.com

[84] NerveCenter, http://www.open.com/htm/nervecenter.htm

[85] SOC Security Operations Center, http://www.mainroad.pt

[86] Blade SmartEvent Software, http://www.checkpoint.com/softwareblades/smartevent.html

[87] Hawk Network Defense Enterprise Event Correlation & Heuristic Trend Analysis for Unified Information Security, http://www.hawkdefense.com

[88] Shane O’Donnell, “Network Management: Open Source Solutions to Proprietary Problems”, Proceedings of the 28th SIGUCCS Conference on User Services, pp. 208-217, October 2000.

[89] CLIPS, http://clipsrules.sourceforge.net/

[90] G. Jakobson and M. Weissman, “Real-time telecommunication network management: Extending event correlation with temporal constraints”, Proceedings of the 4th International Symposium on Integrated Network Management, pp. 290-301, May 1995.

[91] Peter Wainwright et al., Professional Perl Programming, Birmingham, UK, Wrox Press Ltd., 2001.

[92] Hugh R. Casey, The Simple Event Monitor, A Tool for Network Management, MSc thesis, University of Colorado, 2002.

[93] Jim Brown, Working with SEC- the Simple Event Correlator, Copyright © 2003 by Jim Brown, November 23, 2003

[94] Jim Brown, Working with SEC- the Simple Event Correlator Part Two, Copyright © 2003 by Jim Brown, November 23, 2003

[95] The Perl Programming Language, http://www.perl.org/

[96] Perl Regular Expressions, http://search.cpan.org/~nwclark/perl-5.8.2/pod/perlretut.pod

[97] Perl Regular expressions, http://www.troubleshooters.com/codecorn/littperl/perlreg.h

[98] Perl Regular Expressions, http://www.cs.tut.fi/~jkorpela/perl/regexp.html

[99] Snmptrapd, http://www.net-snmp.org/docs/man/snmptrapd.html

[100] Snmpd, http://www.net-snmp.org/docs/man/snmpd.html

[101] Ubuntu manuals, http://manpages.ubuntu.com/

[102] Snmptrapd.conf, http://manpages.ubuntu.com/manpages/dapper/man8/snmptrapd.8.h

[103] Masum Hasan, Lawrence Ho, Frank Feather, and Binay Sugla. The software frameworks for network management event correlation and filtering systems, Bell Laboratories, Lucent Technologies, 101 Crawfords Corner Road, Holmdel, NJ 07733, USA, 1999.

[104] G. Jakobson and M. Weissman. Real-Time Telecommunication Network Management: Extending Event Correlation with Temporal Constraints. IntegratedNetwork Management IV, IEEE Press, 1995.

[105] Masum Hasan. An active temporal model for network management databases. In A.S. Sethi, Y. Raynaud, and F. Faure-Vincent, editors, Proceedings of the IEEE/IFIP Fourth Intemational Symposium on Integrated Network Management 1995.

[106] S. Klinger, S., Yemini, S., Yemini, Y., Ohsie, D. & Stolfo. A Coding Approach to Event Correlation, In Proceedings of the Fourth Symposium on Integrated Network Management, Santa Barbara, California, USA, Chapman & Hall, pp. 266—277, 1995.

[107] Kenneth R. Sheers. HP OpenView Event Correlation Services. Hewlett-Packard Journal, October 1996.

[108] D. Ohsie, A. Mayer, S. Klinger, and S. Yemini, “Event modeling MODEL language,” in Proceedings of the Fifth IFIP/IEEE International Symposium on Integrated Network Management, pp. 625–637, 1997.

[109] M. Hasan, B. Sugla and R. Viswanathan, "A Conceptual Framework for Network Management Event Correlation and Filtering Systems", in Proc. 6th IFIP/IEEE International Symposium on Integrated Network Management (IM 1999), Boston, MA, USA, May 1999, pp. 233–246.

[110] Networking Academy, OSI Model, Cisco ICND1, ICND2, Cisco Public 2007.

[111] Hélder Vieira Mendes, Gestão e Monitorização de uma rede Departamental – Um caso de Estudo, Departamento de Engenharia Electrotécnica, Instituto Superior de Engenharia do Porto 2008

[112] João Neves, SNMPv1 The Simple Network Management Protocol - version 1, Faculdade de Engenharia do Porto, 2006

[113] João Neves, SNMPv2 A Evolução do SNMP, Faculdade de Engenharia do Porto, 2006

[114] João Neves, Gestão de Redes, Faculdade de Engenharia do Porto, 2006

[115] Stallings, William. SNMP, SNMPv2, and CMIP: The Practical Guide to Network Management Standards. Addison-Wesley, 1993

[116] Stallings, William. SNMP, SNMPv2, and RMON 1 and 2, 1999

[117] Claudionor N. Coelho Jr, Gerênciamento de Redes de Telecomunicações e de Computadores, Departamento de Ciência da Computação – UFMG, 1999

[118] http://www.simpleweb.org/

[119] Cisco, Simple Network Management Protocol, Internet Technologies HandbooK, http://www.cisco.com/en/US/docs/internetworking/technology/handbook/SNMP.html

[120] Beethovem Zanella Dias, Nilton Alves Jr, Protocolo de Gerenciamento SNMP, 2002.

[121] Jeff Parker , FCAPS, TMN & ITIL – Three Key Ingredients to Effective IT Managment, OpenWater Solutions, LLC, May 6, 2006.

[122] A Conceptual Framework for Network Management Event Correlation and Filtering Systems Masum Hasan Binay Sugla Ramesh Viswanathan Bell Laboratories, Lucent Technologies 101 Crawfords Corner Road, 1999.

[123] Risto Vaarandi; Platform Independent Tool for Local Event Correlation, Acta Cybernetica 15 (2002)

[124] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Klemettinen, M.: A Knowledge Discovery Methodology for Telecommunication Network Alarm Databases, PhD thesis, University of Helsinki, Finland, 1999.

[125] Dilmar Malheiros Meira. A Model for Alarm Correlation in Telecommunication Networks. PhD Thesis, Federal University of Minas Gerais, Brazil, November 1997.

[126] FREEMAN, Roger L, Telecommunication System Engineering, 4ªed., John Wiley & Sons, Inc., Hoboken, New Jersey, 2004.

[127] Boris Gruschke, Integrated Evente Management: Event Correlation Usisng Dependency Grafhs, Department of Computer Science, University of Munich Oettingenstr. 67, 80538 Munich, Germany, 1998.

[128] Martin Roesch, SNORT - Lightweight Intrusion Detection for Networks, Proceedings of LISA '99: 13th Systems Administration Conference, 1999.

[129] http://www.snort.org

[130] http://www.openview.hp.com/products/ecs/index.html

[131] Guilherme Peretti Pezzi, Programando Sistemas Baseados no Conhecimento Utilizando o Ambiente CLIPS, Curso de Pós-Graduação em Ciência da Computação, 2005/2

[132] Mr. Robert E. Lemon, Pittsburgh Energy Technology Center, "Application of artaficial Intelligence to Reservoir Caracterization : An Interdisciplinary Aproach, Submitted by The UniverSiry of Tulsa Tulsa, OK 74104, 1995.

[133] http://www.securityfocus.com/tools/

[134] http://www.silicondefense.com/software/snortsnarf/index.htm

[135] ROSE, Marshall T. The Simple Book: An Introduction to Networking Management, Readings in Simple Network Management Protocol. Prentice Hall, Inc, 1996.

[136] J. Schönwälder and J. Quittek: Secure Management By Delegation within the Internet Management Framework, in Proc. 6th IFIP/IEEE International Symposium on Integrated Network Management, Boston, Seite 687-700, Mai 1999 (SQ99).

[137] ZIMMERMANN, Hubert, OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection IEEE Transactions on Commnications, VLO. COM-28, NO4, APRIL 1980.

[138] CISCO, http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/introint.htm

[139] STALLINGS, William – Data and computer communications. 8ª ed. New Jersey. Person Education, 2007.

[140] TANENBAUM, Anfrew – Computer Networks. 4ª ed. New Jersey. Personal Education International. 2003.

[141] NAV, http://metanav.uninett.no.

[142] UNINETT - http://www.uninett.no/index.en.html.

[143] NAV Users - http://metanav.uninett.no/navusers.

[144] NAV / Technical Documentation - http://metanav.uninett.no/navtechdoc

[145] Marc Dacier, Institut Eurecom Corporate Communications Department, FRANCE Research Report RR-03-093 White Paper: Alert Correlation: Review of the state of the art1 November 28, 2003

[146] SnortCenter http://users.telenet.be/larc/index.html

[147] G. Jakobson, M.D. Weissmann. “Alarm Correlation”. IEEE Network, p.52-59. Nov. 1993.

[148] G. Jakobson, M. Weissman, Shri Goyal, “IMPACT: Development and eployment Ex esience of Network Correlation Applications”, GTE Labomories Incorporated 40 Sylvan Road Waltham, MA 02254, USA, 1995.

[149] G. Jakobson, M. Weissman. “Real-time telecommunication network management: extending event correlation with temporal constraints”. In IFIP/IEEE International Symphosium on Integrated Network Management, IV, page 290-301. 1995.

[150] G. jakobson, M. Weissman, L. Brenner, C. Lafond, C. Matheus. “GRACE: Building Next Generation Event Correlation Services”. GTE Laboratories Incorporated, USA, IEEE, 0-7803-5864-3, p.701-714, 2000.

[151] Fabien Pouget, Marc Dacier, "Alert Correlation: Review of the state of the art", Institut Eurecom Corporate Communications Department , FRANCE, Research Report RR-03-093, November 28, 2003.

[152] Risto Vaarandi, Tools and Techniques for Event Log Analysis", Faculty of Information Technology Department of Computer Engineering Chair of System Programming TALLINN UNIVERSITY OF TECHNOLOGY, 2005

[153] Simple Log Watcher, http://sourceforge.net/projects/swatch, http://swatch.sourceforge.net

[154] Swatch, http://www.linuxcertif.com/man/1/swatch/

[155] Swatch (1) - Linux man page, http://linux.die.net/man/1/swatch.

[156] R. Davis, H. Shrobe. and W. Hamscher, "Diagnosis Based on Description of Structure and Function," Proc. 1982 Nat'l. Conf. Artificial Intelligence, Pittsburgh, Pa., pp.137142. 1982.

[157] James E. Prewett. Analyzing cluster log files using Logsurfer, The Center for High Performance Computing at UNM, June 2003. Available from World Wide Web: http://www.linuxclustersinstitute.

[158] LogSurfer, http://www.crypt.gen.nz/logsurfer/

[159] LogSurfer, http://sourceforge.net/projects/logsurfer

[160] Andreas Hanemann, "A Hybrid Rule-Based/Case-Based Reasoning Approach for Service Fault Diagnosis", Munich Network Management Team Leibniz Supercomputing Center Barer Str. 21, D-80333 Munich, Germany, 2005.

[161] Lewis, L. "A case-based reasoning approch to the management of faults in communication networks. Artificial Intelligence for aplications, 1993.

[162] Ryan Wishart, Event Correlation System For an Elvin-based Sensor Network, October 2002

[163] L.Lewis, Event Correlation in SPECTRUM and other commercial products, SPECTRUM, technical note ctron-lml-99-05, May 1999.

[164] Rouvellou, I.; Hart, G.W., "Automatic alarm correlation for fault identification," INFOCOM '95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People. Proceedings. IEEE, vol., no., pp.553-561 vol.2, 2-6 Apr 1995.

[165] Bouloutas, A.T.; Calo, S.; Finkel, A., "Alarm correlation an fault identification in communication networks", "Communications, IEEE Transactionson. vol42, no.234, pp.523-533, Feb/Mar/Apr 1994.

[166] Jaesung Choi; Myungwhan Choi; Sang-Hyuk Lee, "An alarm correlation and fault identification scheme based on OSI managed object classes, "Communications, 1999. ICC'99.1999 IEEE International Conference o, vol 3, npo, pp.1547-1551 vol3, 1999.

[167] Yu, M., Li and Chung, L W. 2006 “A pratical scheme for MPLS fault monitoring and alarm correlation in backbone networks”. Comput. Networks 50, 16 Nov. 2006.

[168] Hélder Mendes, GILT - ISEP, Jorge Mamede, GILT - ISEP, Artigo, Preservação da QOS durante a gestão remota de software, 2008.

Anexo A. Algoritmo estudo com 2 regras de correlação causal de eventos recolhidos do NMSIS.

Neste anexo estão 2 tipos de regras *PairWithWindow* e *Pair* do ficheiro de configuração com um algoritmo de correlação causal com eventos recolhidos do NMSIS com a respectiva explicação em cada regra.

```
# processo de 2 regras com 4 tipos de querys: de servico, servidor, switch e
gateway
# Exemplo C4.2.conf
# Reconhece qq pattern  e escrecve para uma file stream (FIFO).
# processo de querys de serviço, servidor, switch e gateway

type=PairWithWindow
ptype=RegExp
pattern=.*Servidor:\s(\d+)\sIP:\s([\d\.]+)\sTipoServiço:\s(\w+)\sEstado:\sn
desc=serviço $3 servidor $2
action=write SEC NMSIS %t Falha no serviço $3 do servidor $2
continue2=TakeNext
ptype2=RegExp
pattern2=.*ElementoRede:\s$1\sCategoria:\s(\w+)\sIP:\s([\d\.]+)\sSalaID:\s(\w+)
\sEstado:\sn\sMAC:\s(\w+:\w+:\w+:\w+:\w+:\w+)\sNomeSwitch:\s([\d\.]+)\sPorto:\s
(\d+)
desc=elemento %1 ip $2 switch $5 porto $6
action2=write SEC NMSIS %t Falha no serviço %3 devido à ligação Servidor $2 e
Switch $5 Porto $6
window=40

1ª Regra:
*A regra PairWithWindow, contem 2 patterns, 2 acções e uma janela de tempo. A
action2 (2° acção) é executada se e só se os eventos A e B ocorrerrem dentro da
janela. Se A ocorrer e B não ocorrer, apenas a action (1° acção) é executada
após o tempo da janela.

* Se a pattern
[.*Servidor:\s(\d+)\sIP:\s([\d\.]+)\sTipoServiço:\s(\w+)\sEstado:\sn] é
recebida seguida de outra pattern
[.*ElementoRede:\s$1\sCategoria:\s(\w+)\sIP:\s([\d\.]+)\sSalaID:\s(\w+)\sEstado
:\sn\sMAC:\s(\w+:\w+:\w+:\w+:\w+:\w+)\sNomeSwitch:\s([\d\.]+)\sPorto:\s(\d+)]
dentro de 40 segundos, já que o pooling é actualizado de 30 em 30 segundos,com
o mesmo valor da variavel ($1) que é o valor da coluna netboxid utlilizado nas
2 querys, é escrito na saída num ficheiro de log (ou pipe), com data e hora
interna do SEC [Falha no serviço %3 devido à ligação Servidor $2 e Switch $5
Porto $6 ] e é criado um novo evento interno [elemento %1 ip $2 switch $5 porto
$6] e passado às outras regras.

* Se a 2ª pattern não for recebida  dentro de 40 segundos é escrito num
ficheiro de log (ou pipe) ["Falha no serviço $3 do servidor $2"] e criado um
novo evento interno [serviço $3 servidor $2] e passado às outras regras.

type=Pair
ptype=RegExp
pattern=.*Servidor:\s(\d+)\sIP:\s([\d\.]+)\sTipoServiço:\s(\w+)\sEstado:\sn
desc=serviço $3 servidor $2
action=write SEC_NMSIS %t Falha no serviço $3 do servidor $2
continue2=TakeNext
ptype2=RegExp
pattern2=.*ElementoRede:\s$1\sCategoria:\s(\w+)\sIP:\s([\d\.]+)\sSalaID:\s(\w+)
\sEstado:\sn\sMAC:\s(\w+:\w+:\w+:\w+:\w+:\w+)\sNomeSwitch:\s([\d\.]+)\sPorto:\s
(\d+)
desc=elemento %1 ip $2 switch $5 porto $6
action2=write SEC NMSIS %t Falha no serviço %3 devido à ligação Servidor $2 e
Switch $5 Porto $6
window=40
```

```
* A regra Pair trata de 2 patterns, 2 accções e uma janela de tempo dentro da
mesma regra, utiliza uma janela temporal que é activada após a primeira
ocorrência do evento A. Se o evento B ocorre dentro da janela, os eventos A e B
são considerados correlacionados e o matching é considerado à regra inteira. Se
não, a operação de correlaçao para o par (A e B) termina. A action (1° acção) é
executada de imediato quando é feito o matching a A e a action2 (2° acção) é
executada quando é feito o matching do evento B na mesma janela temporal.

* Faz matching a pattern
[.*Servidor:\s(\d+)\sIP:\s([\d\.]+)\sTipoServiço:\s(\w+)\sEstado:\sn], se for
recebida, escreve num ficheiro de log (ou pipe), um alarme com o seguinte
contexto com data e hora interna do SEC ["Falha no serviço $3 do servidor $2"]
é gerado imediatamente e um event interno [serviço $3 servidor $2] é criado.

*Se a 2ª pattern não for recebida dentro de 40 segundos, não faz mais nenhuma
acção.
```

Anexo B. Algoritmo final com 3 regras de correlação causal de eventos recolhidos do NMSIS

Neste anexo estão as três regras do ficheiro de configuração com um algoritmo de correlação causal de eventos recolhidos do NMSIS.

```
# processo de 3 regras com 4 tipos de querys: de servico, servidor, switch e
gateway

type=PairWithWindow
ptype=RegExp
pattern=.*IP:\s([\d\.]+)\sTipoServico:\s(\w+)\sEstado:\sn
desc=servico $2 servidor $1 down1
action=event %s; \
write trap.txt %t Falha no servico $2 do servidor $1;
continue2=TakeNext
ptype2=RegExp
pattern2=.*SRV\sIP:\s$1\sSalaID:\s(\w+)\sEstado:\sn\s.*Elemento:\s([\d\.]+)\sPo
rto:\s(.*)
desc2=servico %2 servidor %1 switch $2 porto $3 down2
action2=event %s;
window=20

type=PairWithWindow
ptype=RegExp
pattern=servico (\S+) servidor (\S+) switch (\S+) porto (\S.*) down2
desc=servico $1 servidor $2 switch $3 porto $4 down3
action=event %t %s; \
write trap.txt %t Falha no servico $1 devido a¡ quebra na ligacao entre
Servidor $2 e Switch $3 Porto $4;
ptype2=RegExp
pattern2=.*SW\sIP:\s$3\sSalaID:\s(\w+)\sEstado:\sn\s.*Elemento:\s([\d\.]+)\sPor
to:\s(.*)
desc2=servico %1 servidor %2 switch %3 gateway $2 porto $3 down4
action2=event %t %s;
window=20

type=PairWithWindow
ptype=RegExp
pattern=servico (\S+) servidor (\S+) switch (\S+) gateway (\S+) porto (\S+)
down4
desc=servico $1 servidor $2 switch $3 gateway $4 porto $5 down5
action=event %t %s; \
write trap.txt %t Falha no servico $1 do Servidor $2 devido a quebra na ligacao
entre Switch $3 e Gateway $4 Porto $5;
ptype2=RegExp
pattern2=.*GW\sIP:\s$4\sSalaID:\s(\w+)\sEstado:\sn
desc2=servico $1 servidor $2 switch $3 gateway $4 porto $5 down6
action2=event %t %s; \
write trap.txt %t Falha no servico %1 do Servidor %2 devido a Gateway $2 Porto
$3 isoldada da rede
window=20
```

Anexo C. Algoritmo de 4 regras de correlação temporal de eventos recolhidos do snmptrapd

Neste anexo estão quatro regras do ficheiro de configuração com um algoritmo de correlação temporal de eventos recolhidos via NET-SNMP (snmptrap) com a respectiva explicação em cada regra.

```
# processo linkDown/linkUp  notificação de eventos recebidos via NET-SNMP

type=PairWithWindow
ptype=RegExp
pattern=(\d+-\d+-
\d+)\s(\d+:\d+:\d+)\s((\d+\.\d+\.\d+\.)\d+).*STRING:\s([\w\/]+)\s.*STRING:\s\"d
own"
desc=ALARME_TRAP $1 $2 Host $3 Net $4 $5 estado DOWN
action=event %s;
continue2=TakeNext
ptype2=RegExp
pattern2=\s$3.*STRING:\s$5\s.*STRING:\s\"up"
desc2=ALARME_TRAP Host %3 Net %4 Port %5 estado INTERMITENTE
action2=event %t %s ;
window=20

1ª Regra:
*A regra PairWithWindow, contem 2 patterns, 2 acções e uma janela de tempo. A
action2 (2° acção) é executada se e só se os eventos A e B ocorrerrem dentro da
janela. Se A ocorrer e B não ocorrer, apenas a action (1° acção) é executada
após o tempo da janela.
* Se a pattern [xxxx-xx-xx xx:xx:xx xxx.xxx.xxx.xxx STRING: x STRING: "down"] é
recebida seguida de outra pattern [xxxx-xx-xx xx:xx:xx xxx.xxx.xxx.xxx STRING:
x STRING: "up"] dentro de 20 segundos, com o mesmo valor (xxx.xxx.xxx.xxx
STRING: x) é considerado INTERMITENTE (intermitente) e é criado um novo evento
interno [CISCO Host xxx.xxx.xxx.xxx x state INTERMITENTE] e passado às outras
regras , com data e hora interna do SEC.
* Se a pattern [xxxx-xx-xx xx:xx:xx xxx.xxx.xxx.xxx STRING: x STRING: "down"] é
recebida e não é seguida de outra pattern [xxxx-xx-xx xx:xx:xx xxx.xxx.xxx.xxx
STRING: x STRING: "up"] dentro de 20 segundos é considerado DOWN e é criado um
novo evento interno [CISCO xxxx-xx-xx xx:xx:xx Host xxx.xxx.xxx.xxx x state
DOWN]e passado às outras regras, com data e hora da trap de origem.

type=SingleWithSuppress
continue=TakeNext
ptype=RegExp
pattern=ALARME TRAP (\S+) (\S+) Host (\S+) Net (\S+) (\S+) estado DOWN
desc=EVENTO TRAP SUPRIME estado DOWN
action=reset +1 %s;
window=60


2ª Regra:
* A regra SingleWithSuppress, contem uma pattern, eventos que façam matching
são suprimidos.A primeira vez que o evento é visto fica alerta e ignora esse
mesmo evento durante uma intervalo de tempo.
* Faz matching a pattern gerada internamente [CISCO xxxx-xx-xx xx:xx:xx Host
xxx.xxx.xxx.xxx x state DOWN] e permite que esta mensagem seja processada
apenas 1 vez em 60 segundos. Cancela qualquer evento de operação de correlação
que tenha esta pattern durante 60 segundos antes de passar a proxima regra (+1
é | 2 | a 3ª regra, regras de 0-N, "Cancelling the correlation operation with
key 'C3.4.conf | 2 | cisco xxxx-xx-xx xx:xx:xx Host xxx.xxx.xxx.xxx x state
down).


type=Pair
ptype=RegExp
```

```
pattern=ALARME_TRAP (\S+) (\S+) Host (\S+) Net (\S+) (\S+) estado DOWN
desc=alarme trap $1 $2 host $3 net $4 $5 estado down
action=event %s; \
write SEC NMSIS %t Host $3 Net $4 $5 MAJOR DOWN;
ptype2=RegExp
pattern2=\s$3.*STRING:\s$5\s.*STRING:\s\"up"
desc2=alarme trap host %3 %5 state up
action2=write SEC NMSIS %t Host %3 Net %4 %5 NORMAL UP
window=86400
```

3ª Regra:
* A regra Pair trata de 2 patterns, 2 accções e uma janela de tempo dentro da mesma regra, utiliza uma janela temporal que é activada após a primeira ocorrência do evento A. Se o evento B ocorre dentro da janela, os eventos A e B são considerados correlacionados e o matching é considerado à regra inteira. Se não, a operação de correlaçao para o par (A e B) termina. A action (1° acção) é executada de imediato quando é feito o matching a A e a action2 (2° acção) é executada quando é feito o matching do evento B na mesma janela temporal.
* Faz matching a pattern gerada internamente [CISCO xxxx-xx-xx xx:xx:xx Host xxx.xxx.xxx.xxx x state DOWN] se for recebida, escreve num ficheiro de log (ou pipe) um alarme com o seguinte contexto com data e hora interna do SEC [Host xxx.xxx.xxx.xxx x state major down] é gerado imediatamente.
* Faz matching a pattern [xxxx-xx-xx xx:xx:xx xxx.xxx.xxx.xxx STRING: x STRING: "up"] e se tiver o mesmo valor (xxx.xxx.xxx.xxx STRING: x) da string [xxxx-xx-xx xx:xx:xx xxx.xxx.xxx.xxx STRING: x STRING: "down"] dentro de um periodo de 24 horas, escreve num ficheiro de log (ou pipe) um alarme com o seguinte contexto com data e hora interna do SEC [Host xxx.xxx.xxx.xxx x state normal up].
* A janela de correlação de cada interface é feita diariamente por um periodo de 24 horas, findo esse tempo sem haver correlação a operação termina.

```
type=SingleWith2Thresholds
ptype=RegExp
pattern=ALARME_TRAP Host (\S+) Net (\S+) (\S+) estado INTERMITENTE
desc=alarme trap $1 $2 host $3 $5 estado instavel
action=write SEC NMSIS %t Host $1 Net $2 $3 MAJOR LIGACAO INSTAVEL
window=40
thresh=2
desc2=alarme_trap $1 $2 Host $3 $5 novamente estavel
action2=write SEC_NMSIS %t Host $1 Net $2 $3 NORMAL ESTAVEL
window2=7200
thresh2=0
```

4ª Regra:
* A regra SingleWithThreshold é usada para contar o número de eventos com match (correspondência) dentro de uma janela temporal. Se o numero de match exceder o treshold a action (1° acção) é executada. Se o numero de eventos com matching ñ exceder o treshold dentro da janela, a janela "slides" ou seja, o tempo de começo para a janela de correlação é movido para a segunda ocorrencia de matching pattern. Este processo repete-se até ao tempo da janela expirar sem nenhum novo evento com matching.
*Se 2 pattern geradas internamente [CISCO Host xxx.xxx.xxx.xxx x state INTERMITENTE] forem detectadas durante um periodo de 40 min, escreve num ficheiro de log (ou pipe) um alarme com o seguinte contexto com data e hora interna do SEC [Host xxx.xxx.xxx.xxx x state major unstable].
* Se após o ultimo alerta de "major unstable não existir mais nenhum matching com evento interno [CISCO Host xxx.xxx.xxx.xxx x state INTERMITENTE] durante um periodo de 2 horas, escreve num ficheiro de log (ou pipe) um alarme com o seguinte contexto com data e hora interna do SEC [Host xxx.xxx.xxx.xxx x state normal state].

Anexo D. Exemplo de "*pattern de matching*" em vários eventos via query ao NMSIS, recorrendo as expressões regulares.

Neste anexo estão exemplos dos quatro tipos de querys e as respectivas expressões regulares responsáveis pela sua correspondência.

```
# Query Serviço
# Query * = Evento do NMSIS

Query Servico IP: 10.1.33.1 TipoServico: http Estado: n

.*IP:\s([\d\.]+)\sTipoServico:\s(\w+)\sEstado:\sn

# Query Servidor

Query SRV IP: 10.0.32.71 SalaID: F507 Estado: y CONECTADO  Elemento:
10.0.32.100 Porto: FastEthernet0/8

Query SRV IP: 10.1.33.1 SalaID: F507 Estado: y CONECTADO  Elemento: 10.1.33.100
Porto: BayStack 450-24T - 1

.*SRV\sIP:\s((\d+\.\d+\.\d+\.)\d+)\sSalaID:\s(\w+)\sEstado:\sn\s.*Elemento:\s([
\d\.]+)\sPorto:\s(.*)

# Query Switch

Query SW IP: 10.1.33.100 SalaID: F507 Estado: y CONECTADO  Elemento:
10.0.32.250 Porto: FastEthernet0/1

Query SW IP: 10.0.32.100 SalaID: F507 Estado: y CONECTADO  Elemento:
10.0.32.250 Porto: FastEthernet0/0

.*SW\sIP:\s((\d+\.\d+\.\d+\.)\d+)\sSalaID:\s(\w+)\sEstado:\sn\s.*Elemento:\s([\
d\.]+)\sPorto:\s(.*)


# Query Gateway

Query GW IP: 10.0.32.250 SalaID: F507 Estado: y

.*GW\sIP:\s((\d+\.\d+\.\d+\.)\d+)\sSalaID:\s(\w+)\sEstado:\sn
```

Anexo E. Exemplos de "*pattern de matching*" de várias traps recorrendo as expressões regulares.

Neste anexo estão vários tipos de traps com as respectivas expressões regulares responsáveis pela correspondência.

```
# trap do SW Cisco

2010-07-26 21:15:55 10.0.32.100(via UDP: [10.0.32.100]:50465) TRAP, SNMP v1,
community public SNMPv2-SMI::enterprises.9.1.217 Link Down Trap (0) Uptime: 1
day, 7:01:17.63 IF-MIB::ifIndex.9 = INTEGER: 9  IF-MIB::ifDescr.9 = STRING:
FastEthernet0/8     IF-MIB::ifType.9 = INTEGER: ethernetCsmacd(6)
SNMPv2-SMI::enterprises.9.2.2.1.1.20.9 = STRING: "down"


(\d+-\d+-\d+)\s(\d+:\d+:\d+)\s((\d+\.\d+\.\d+\.)\d+).*STRING:\s([\w\/]+)\s.
*STRING:\s\"down"

# trap do SW Nortel

2010-07-26 20:42:48 10.1.33.100(via UDP: [10.1.33.100]:161) TRAP, SNMP v1,
community public SNMPv2-SMI::enterprises.45.3.35.1 Link Down Trap (0) Uptime: 1
day, 6:14:33.70 IF-MIB::ifIndex.4 = INTEGER: 4
2010-07-26 20:43:21 10.1.33.100(via UDP: [10.1.33.100]:161) TRAP, SNMP v1,
community public SNMPv2-SMI::enterprises.45.3.35.1 Link Up Trap (0) Uptime: 1
day, 6:14:41.95 IF-MIB::ifIndex.4 = INTEGER: 4

(\d+-\d+-\d+)\s(\d+:\d+:\d+)\s((\d+\.\d+\.\d+\.)\d+).*Link\s([\w\s\w]+)\s
(\d+-\d+-\d+)\s(\d+:\d+:\d+)\s((\d+\.\d+\.\d+\.)\d+).*Link\s([\w\s\w]+)\s.
*INTEGER:\s\(\d+)

# trap do GW Cisco

2010-07-26 20:44:08 0.0.0.0(via UDP: [10.0.32.250]:6563) TRAP, SNMP v1,
community public SNMPv2-SMI::enterprises.9.1.209 Link Down Trap (0) Uptime: 1
day, 6:29:13.75 IF-MIB::ifIndex.2 = INTEGER: 2  IF-MIB::ifDescr.2 = STRING:
FastEthernet0/1     IF-MIB::ifType.2 = INTEGER: ethernetCsmacd(6)     SNMPv2-
SMI::enterprises.9.2.2.1.1.20.2 = STRING: "Keepalive failed"

2010-07-26 20:44:41 0.0.0.0(via UDP: [10.0.32.250]:6563) TRAP, SNMP v1,
community public SNMPv2-SMI::enterprises.9.1.209 Link Down Trap (0) Uptime: 1
day, 6:29:14.75 IF-MIB::ifIndex.2 = INTEGER: 2  IF-MIB::ifDescr.2 = STRING:
FastEthernet0/1     IF-MIB::ifType.2 = INTEGER: ethernetCsmacd(6)     SNMPv2-
SMI::enterprises.9.2.2.1.1.20.2 = STRING: "Lost Carrier"

2010-07-26 20:45:47 0.0.0.0(via UDP: [10.0.32.250]:6563) TRAP, SNMP v1,
community public SNMPv2-SMI::enterprises.9.1.209 Link Up Trap (0) Uptime: 1
day, 6:29:23.78 IF-MIB::ifIndex.2 = INTEGER: 2  IF-MIB::ifDescr.2 = STRING:
FastEthernet0/1     IF-MIB::ifType.2 = INTEGER: ethernetCsmacd(6)     SNMPv2-
SMI::enterprises.9.2.2.1.1.20.2 = STRING: "Keepalive OK"

(\d+-\d+-\d+)\s(\d+:\d+:\d+)\s((\d+\.\d+\.\d+\.)\d+).*STRING:\s([\w\/]+)\s.
*STRING:\s\"Keepalive failed"
(\d+-\d+-\d+)\s(\d+:\d+:\d+)\s((\d+\.\d+\.\d+\.)\d+).*STRING:\s([\w\/]+)\s.
*STRING:\s\"Lost Carrier"

desc=ALARME TRAP $1 $2 Host $3 Net $4 $5 estado Keepalive failed
desc=ALARME TRAP $1 $2 Host $3 Net $4 $5 estado Lost Carrier
pattern2=\s$3.*STRING:\s$5\s.*STRING:\s\"Keepalive OK"
```

Anexo F. Script PHP de query ao NMSIS

Neste anexo está os quatro tipos de querys ao NMSIS e gera 4quatro tipos de eventos de saída.

```
<?php

while (true){


               $cstrNF = "host=10.0.32.72 user=NMSIS dbname=manage
password=t44otypecS2y";
            echo "String de ligacao: " . $cstrNF;

            if (! $thisNF->dbh_dbcon = pg_connect($cstrNF) ) {
                print "<h1>" . gettext("Database error") . "</h1>";
                print "<p>" . gettext("Could not connect to the manage
database. The database server could be down, or the logininfo could be corrupt
in the database configuration file.");
                exit(0);
            }
                        else print "\n" . gettext("Criou ligacao a Base de Dados
MANAGE do NMSIS!") . "\n";

                        $query="SELECT netbox.ip AS ip1,service.handler AS
handler2,service.up AS up2  FROM netbox, service WHERE netbox.netboxid =
service.netboxid";
                        $resultado=pg_query($query);

                        while ($row = pg fetch array($resultado, NULL,
PGSQL ASSOC)){

                                $ip = $row['ip1'];
                                $handler = $row['handler2'];
                                $up = $row['up2'];

                                echo"Query ServiÃ§o IP: $ip TipoServiÃ§o:
$handler Estado: $up\n";
                                $filename = 'test.txt';
                                $somecontent = "Query ServiÃ§o IP: $ip
TipoServiÃ§o: $handler Estado: $up\n";

                                // Let's make sure the file exists and is
writable first.
                                if (is_writable($filename)) {

                                        if (!$handle = fopen($filename, 'a')) {
                                        echo "Cannot open file ($filename)\n";
                                        exit;
                                        }

                                        if (fwrite($handle, $somecontent) ===
FALSE) {
                                        echo "Cannot write to file ($filename)\n";
                                exit;
                                        }

                                echo "\n Success, wrote ($somecontent) to file
($filename)\n";

                                fclose($handle);

                                }
                                else {
                                echo "The file $filename is not writable";
                                }
```

```
                    }



               $cstrNF = "host=10.0.32.72 user=NMSIS dbname=manage
password=t44otypecS2y";
            echo "String de ligacao: " . $cstrNF;

            if (! $thisNF->dbh dbcon = pg connect($cstrNF) ) {
                print "<h1>" . gettext("Database error") . "</h1>";
                print "<p>" . gettext("Could not connect to the manage
database. The database server could be down, or the logininfo could be corrupt
in the database configuration file.");
                exit(0);
            }
                    else print "\n" . gettext("Criou ligacao a Base de Dados
MANAGE do NMSIS!") . "\n";

                    $query="SELECT netbox.netboxid AS netboxid1,netbox.catid
AS catid1,netbox.roomid AS roomid1,netbox.up AS up1,arp.ip AS ip2,arp.mac AS
mac2,cam.sysname AS sysname3,cam.port AS port3  FROM netbox, arp, cam WHERE
arp.mac=cam.mac AND netbox.ip = arp.ip AND arp.end_time ='infinity'";
                    $resultado=pg_query($query);

                    while ($row = pg fetch array($resultado, NULL,
PGSQL ASSOC)){

                            $netboxid = $row['netboxid1'];
                            $catid = $row['catid1'];
                            $ip = $row['ip2'];
                            $roomid = $row['roomid1'];
                            $up = $row['up1'];
                            $mac = $row['mac2'];
                            $to_sysname = $row['sysname3'];
                            $to port = $row['port3'];

                            echo"Query $catid IP: $ip SalaID: $roomid Estado:
$up CONECTADO  Elemento: $to_sysname Porto: $to_port\n";
                            $filename = 'test.txt';
                            $somecontent = "Query $catid IP: $ip SalaID:
$roomid Estado: $up CONECTADO  Elemento: $to sysname Porto: $to port\n";

                            // Let's make sure the file exists and is
writable first.
                            if (is_writable($filename)) {

                                    if (!$handle = fopen($filename, 'a')) {
                                    echo "Cannot open file ($filename)\n";
                                    exit;
                                    }

                                    if (fwrite($handle, $somecontent) ===
FALSE) {
                                    echo "Cannot write to file ($filename)\n";
                            exit;
                                    }

                            echo "\n Success, wrote ($somecontent) to file
($filename)\n";

                            fclose($handle);

                            }
                            else {
                            echo "The file $filename is not writable";
                            }

                    }


               $cstrNF = "host=10.0.32.72 user=NMSIS dbname=manage
password=t44otypecS2y";
            echo "String de ligacao: " . $cstrNF;

            if (! $thisNF->dbh dbcon = pg connect($cstrNF) ) {
                print "<h1>" . gettext("Database error") . "</h1>";
```

```
                print "<p>" . gettext("Could not connect to the manage
database. The database server could be down, or the logininfo could be corrupt
in the database configuration file.");
                exit(0);
            }
                        else print "\n" . gettext("Criou ligacao a Base de Dados
MANAGE do NMSIS!") . "\n";

                        $query="SELECT netbox.catid AS catid1,netbox.roomid AS
roomid1,netbox.up AS up1,arp.ip AS ip2,arp.sysname AS sysname2,gwport.interface
AS interface4 FROM netbox, arp, gwportprefix, gwport WHERE netbox.ip = arp.ip
AND arp.prefixid = gwportprefix.prefixid AND gwportprefix.gwportid =
gwport.gwportid AND netbox.catid ='SW'";
                        $resultado=pg query($query);

                        while ($row = pg_fetch_array($resultado, NULL,
PGSQL_ASSOC)){

                                $catid = $row['catid1'];
                                $ip = $row['ip2'];
                                $roomid = $row['roomid1'];
                                $up = $row['up1'];
                                $to_sysname = $row['sysname2'];
                                $to port = $row['interface4'];

                                echo"Query $catid IP: $ip SalaID: $roomid Estado:
$up CONECTADO  Elemento: $to_sysname Porto: $to_port\n";
                                $filename = 'test.txt';
                                $somecontent = "Query $catid IP: $ip SalaID:
$roomid Estado: $up CONECTADO  Elemento: $to sysname Porto: $to port\n";

                                // Let's make sure the file exists and is
writable first.
                                if (is_writable($filename)) {

                                        if (!$handle = fopen($filename, 'a')) {
                                        echo "Cannot open file ($filename)\n";
                                        exit;
                                        }

                                        if (fwrite($handle, $somecontent) ===
FALSE) {
                                        echo "Cannot write to file ($filename)\n";
                                exit;
                                        }

                                echo "\n Success, wrote ($somecontent) to file
($filename)\n";

                                fclose($handle);

                                }
                                else {
                                echo "The file $filename is not writable";
                                }

                        }



               $cstrNF = "host=10.0.32.72 user=NMSIS dbname=manage
password=t44otypecS2y";
            echo "String de ligacao: " . $cstrNF;

            if (! $thisNF->dbh dbcon = pg connect($cstrNF) ) {
                print "<h1>" . gettext("Database error") . "</h1>";
                print "<p>" . gettext("Could not connect to the manage
database. The database server could be down, or the logininfo could be corrupt
in the database configuration file.");
                exit(0);
            }
                        else print "\n" . gettext("Criou ligacao a Base de Dados
MANAGE do NMSIS!") . "\n";

                        $query="SELECT
netbox.catid,netbox.ip,netbox.roomid,netbox.up FROM netbox WHERE netbox.catid
='GW'";
```

```
                        $resultado=pg_query($query);

                        while ($row = pg fetch array($resultado, NULL,
PGSQL ASSOC)){

                                $catid = $row['catid'];
                                $ip = $row['ip'];
                                $roomid = $row['roomid'];
                                $up = $row['up'];


                                echo"Query $catid IP: $ip SalaID: $roomid Estado:
$up\n";
                                $filename = 'test.txt';
                                $somecontent = "Query $catid IP: $ip SalaID:
$roomid Estado: $up\n";

                                // Let's make sure the file exists and is
writable first.
                                if (is writable($filename)) {

                                        if (!$handle = fopen($filename, 'a')) {
                                        echo "Cannot open file ($filename)\n";
                                        exit;
                                        }

                                        if (fwrite($handle, $somecontent) ===
FALSE) {
                                        echo "Cannot write to file ($filename)\n";
                                exit;
                                        }

                                echo "\n Success, wrote ($somecontent) to file
($filename)\n";

                                fclose($handle);

                                }
                                else {
                                echo "The file $filename is not writable";
                                }

                        }


usleep(30000000); //30000000 us, 30 s
}
?>
```

Anexo G. Script Perl para escrita no pipe do SEC.

Neste anexo um script em Perl que lê os dados vindos do daemon Net-SNMP (snmptrapd) que ve coloca cada trap numa única linha para ser feito o matching pelo correlador.

```
#
# Examplo S3.4.pl - Script para ler e escrever dados num pipe (fifo).
#
use IO::Handle;
$|= 1;
$filename1 = 'trap.txt';
$filename2 = 'sec.txt';
open(LOGIN,  "+< $filename1") or die "login  $!";
open(LOGOUT, ">$filename2") or die "logout  $!";
LOGOUT->autoflush(1);

print "Start of S3.4.04.pl.  Reading $filename1...\n";

$alarm = undef;
while (1) {
while (<LOGIN>)
{
chomp(my $line = $_);
  if (/(^(\d+-\d+-\d+).*)/) {
    print LOGOUT "c1: $alarm\n "if ($alarm);
    $alarm = $line;
  }
  else {
    $alarm .= $line;
  }

}
print "c2: $alarm\n";
}
```

Anexo H. Página WEB de leitura dos eventos de saída do SCEN.

Neste anexo mostra uma página HTML com um script em PHP que lê os dados vindos da pipe ou de um ficheiro de texto de saída do SCEN e coloca linha-a-linha as mensagens numa página WEB e faz o refesh a cada 5 segundos.


```
#
# Exemplo S3.4.pl - Script para ler e escrever dados num pipe (fifo) embebido
# Para ser embebido numa página HTML.
#

<?php
$handle = @fopen("/home/casilva/pessoal/PHP/test.txt", "r");
if ($handle) {
    while (($buffer = fgets($handle, 4096)) !== false) {
        echo $buffer;
       echo "<br>";
    }
    if (!feof($handle)) {
        echo "Error: unexpected fgets() fail\n";
    }
    fclose($handle);
}
?>
```

Anexo I. Algoritmo de teste com 6 regras e junção de correlação temporal e causal.

Neste anexo estão seis regras do ficheiro de configuração com um algoritmo de correlação causal e temporal de eventos recolhidos do NMSIS e via Net-SNMP (snmptrap).

```
type=PairWithWindow
ptype=RegExp
pattern=(\d+-\d+-
\d+)\s(\d+:\d+:\d+)\s((\d+\.\d+\.\d+\.)\d+).*STRING:\s([\w\/]+)\s.*STRING:\s\"d
own"
desc=ALARME TRAP $1 $2 Host $3 Net $4 $5 estado DOWN
action=event %s;
continue2=TakeNext
ptype2=RegExp
pattern2=\s$3.*STRING:\s$5\s.*STRING:\s\"up"
desc2=ALARME TRAP Host %3 Net %4 %5 estado INTERMITENTE
action2=event %t %s ;
window=20

type=SingleWithSuppress
continue=TakeNext
ptype=RegExp
pattern=ALARME TRAP (\S+) (\S+) Host (\S+) Net (\S+) (\S+) estado DOWN
desc=EVENTO_TRAP SUPRIME estado DOWN
action=reset +1 %s;
window=20

type=Pair
ptype=RegExp
pattern=ALARME_TRAP (\S+) (\S+) Host (\S+) Net (\S+) (\S+) estadoDOWN
desc=alarme_trap $1 $2 host $3 net $4 $5 estado down
action=event %s; \
write SEC NMSIS %t host $3 net $4 $5 MAJOR DOWN;
ptype2=RegExp
pattern2=\s$3.*STRING:\s$5\s.*STRING:\s\"up"
desc2=alarme_trap host %3 %5 state up
action2=write SEC_NMSIS %t host %3 net %4 %5 NORMAL UP
window=86400


type=SingleWith2Thresholds
ptype=RegExp
pattern=ALARME TRAP Host (\S+) Net (\S+) (\S+) estado INTERMITENTE
desc=alarme trap $1 $2 host $3 $5 estado instavel
action=write SEC NMSIS %t host $1 net $2 $3 MAJOR LIGAÇAO INSTAVEL
window=40
thresh=2
desc2=alarme_trap $1 $2 Host $3 $5 novamente estavel
action2=write SEC NMSIS %t host $1 net $2 $3 NORMAL ESTAVEL
window2=60
thresh2=0


type=Pair
ptype=RegExp
pattern=.*ElementoID:\s(\d+)\sCategoria:\s(\w+)\sIP:\s([\d\.]+)\sSalaID:\s(\w+)
\sEstado:\sn
desc=ALARME_NMSIS [netbox] ElementoID: $1 Cat: $2 IP: $3 SalaID: $4 "Elemento
de rede [$3] não acessivel "
action=write SEC NMSIS %t %s
continue2=TakeNext
ptype2=RegExp
pattern2=\s$1.*ServiçoID:\s(\d+)\sActivoTimeout:\s(\w+)\sServiceType:\s(\w+)\sE
stado:\sn
```

```
desc2=ALARME_NMSIS [service] ServiçoID: $1 ServiceType: $3 "Quebra serviço
devido falha no elemento [%3]"
action2=write SEC NMSIS %t %s
window=20

type=Single
ptype=RegExp
pattern=.*ElementoID:\s(\d+)\sServiçoID:\s(\d+)\sActivoTimeout:\s(\w+)\sService
Type:\s(\w+)\sEstado:\sn
desc=ALARME_NMSIS [service] ElementoID $1 ServiçoID: $2 ServiceType: $4 "Quebra
serviço"
action=write SEC NMSIS %t %s
```

Anexo J. Algoritmo de teste com 7 regras e junção de correlação temporal e causal.

Neste anexo estão sete regras do ficheiro de configuração com um algoritmo de correlação causal e temporal de eventos recolhidos do NMSIS e via Net-SNMP (snmptrap).

```
type=PairWithWindow
ptype=RegExp
pattern=(\d+-\d+-
\d+)\s(\d+:\d+:\d+)\s((\d+\.\d+\.\d+\.)\d+).*STRING:\s([\w\/]+)\s.*STRING:\s\"d
own"
desc=ALARME TRAP $1 $2 Host $3 Net $4 $5 estado DOWN
action=event %s;
continue2=TakeNext
ptype2=RegExp
pattern2=\s$3.*STRING:\s$5\s.*STRING:\s\"up"
desc2=ALARME TRAP Host %3 Net %4 %5 estado INTERMITENTE
action2=event %t %s ;
window=20

type=SingleWithSuppress
continue=TakeNext
ptype=RegExp
pattern=ALARME TRAP (\S+) (\S+) Host (\S+) Net (\S+) (\S+) estado DOWN
desc=EVENTO_TRAP SUPRIME estado DOWN
action=reset +1 %s;
window=20

type=Pair
ptype=RegExp
pattern=ALARME_TRAP (\S+) (\S+) Host (\S+) Net (\S+) (\S+) estado DOWN
desc=alarme_trap $1 $2 host $3 net $4 $5 estado down
action=event %s; \
write SEC NMSIS %t host $3 net $4 $5 MAJOR DOWN;
ptype2=RegExp
pattern2=\s$3.*STRING:\s$5\s.*STRING:\s\"up"
desc2=alarme_trap host %3 %5 state up
action2=write SEC_NMSIS %t host %3 net %4 %5 NORMAL UP
window=86400

type=Pair
ptype=RegExp
pattern=alarme_trap (\S+) (\S+) host (\S+) net (\S+) (\S+) estado down
desc=EVENTO TRAP Elemento $3 da rede $4 $5 DOWN "VERIFICA EVENTO NMSIS
[netbox]"
action=write SEC NMSIS %t %s
continue2=TakeNext
ptype2=RegExp
pattern2=.*ElementoID:\s(\d+)\sCategoria:\s(\w+)\sIP:\s(($4)\d+)\sSalaID:\s(\w+
)\sEstado:\sn
desc2=ALARME NMSIS [netbox] Elemento: $1 Cat: $2 Sala: $5 " [$3] não acessivel
devido falha no elemento [%3] interface [%5] identificado"
action2=write SEC_NMSIS %t %s
window=40

type=SingleWith2Thresholds
ptype=RegExp
pattern=ALARME_TRAP Host (\S+) Net (\S+) (\S+) estado INTERMITENTE
desc=alarme_trap $1 $2 host $3 $5 estado instavel
action=write SEC_NMSIS %t host $1 net $2 $3 MAJOR LIGAÇAO INSTAVEL
window=40
thresh=2
desc2=alarme trap $1 $2 Host $3 $5 novamente estavel
action2=write SEC_NMSIS %t host $1 net $2 $3 NORMAL ESTAVEL
window2=60
```

```
thresh2=0


type=Pair
ptype=RegExp
pattern=.*ElementoID:\s(\d+)\sCategoria:\s(\w+)\sIP:\s([\d\.]+)\sSalaID:\s(\w+)
\sEstado:\sn
desc=ALARME NMSIS [netbox] ElementoID: $1 Cat: $2 IP: $3 SalaID: $4 "Elemento
de rede [$3] não acessivel "
action=write SEC_NMSIS %t %s
continue2=TakeNext
ptype2=RegExp
pattern2=\s$1.*ServiçoID:\s(\d+)\sActivoTimeout:\s(\w+)\sServiceType:\s(\w+)\sE
stado:\sn
desc2=ALARME NMSIS [service] ServiçoID: $1 ServiceType: $3 "Quebra serviço
devido falha no elemento [%3]"
action2=write SEC_NMSIS %t %s
window=20

type=Single
ptype=RegExp
pattern=.*ElementoID:\s(\d+)\sServiçoID:\s(\d+)\sActivoTimeout:\s(\w+)\sService
Type:\s(\w+)\sEstado:\sn
desc=ALARME NMSIS [service] ElementoID $1 ServiçoID: $2 ServiceType: $4 "Quebra
serviço"
action=write SEC NMSIS %
```

Anexo K. Algoritmo de teste com 11 regras e junção de correlação temporal e causal.

Neste anexo estão onze regras do ficheiro de configuração com um algoritmo de correlação causal e temporal de eventos recolhidos do NMSIS e via Net-SNMP (snmptrap).

```
# traps Cisco SW com correlação de intermitência e correlação com DB NMSIS
netbox.
# expressão regular:(\d+-\d+-
\d+)\s(\d+:\d+:\d+)\s((\d+\.\d+\.\d+\.)\d+).*STRING:\s([\w\/]+)\s.*STRING:\s\"d
own"

type=PairWithWindow
ptype=RegExp
pattern=(\d+-\d+-
\d+)\s(\d+:\d+:\d+)\s((\d+\.\d+\.\d+\.)\d+).*STRING:\s([\w\/]+)\s.*STRING:\s\"d
own"
desc=ALARME TRAP $1 $2 Host $3 Net $4 $5 estado DOWN
action=event %s;
continue2=TakeNext
ptype2=RegExp
pattern2=\s$3.*STRING:\s$5\s.*STRING:\s\"up"
desc2=ALARME TRAP Host %3 Net %4 Port %5 estado INTERMITENTE
action2=event %t %s ;
window=20

type=SingleWithSuppress
continue=TakeNext
ptype=RegExp
pattern=ALARME_TRAP (\S+) (\S+) Host (\S+) Net (\S+) (\S+) estado DOWN
desc=EVENTO_TRAP SUPRIME estado DOWN
action=reset +1 %s;
window=20

type=Pair
ptype=RegExp
pattern=ALARME_TRAP (\S+) (\S+) Host (\S+) Net (\S+) (\S+) estado DOWN
desc=alarme_trap $1 $2 host $3 net $4 $5 estado down
action=event %s; \
write SEC NMSIS %t Host $3 Net $4 $5 MAJOR DOWN;
ptype2=RegExp
pattern2=\s$3.*STRING:\s$5\s.*STRING:\s\"up"
desc2=alarme_trap host %3 %5 state up
action2=write SEC NMSIS %t Host %3 Net %4 %5 NORMAL UP
window=86400

#Correlação entre trap e DB NMSIS netbox

type=Pair
ptype=RegExp
pattern=alarme trap (\S+) (\S+) host (\S+) net (\S+) (\S+) estado down
desc=EVENTO_TRAP Elemento $3 da rede $4 $5 DOWN "VERIFICA EVENTO_NMSIS
[netbox]"
action=write SEC_NMSIS %t %s
continue2=TakeNext
ptype2=RegExp
pattern2=.*ElementoID:\s(\d+)\sCategoria:\s(\w+)\sIP:\s(($4)\d+)\sSalaID:\s(\w+
)\sEstado:\sn
desc2=ALARME_NMSIS [netbox] Elemento: $1 Cat: $2 Sala: $5 " [$3] não acessivel
devido falha no elemento [%3] interface [%5] identificado"
action2=write SEC NMSIS %t %s
window=40

type=SingleWith2Thresholds
ptype=RegExp
```

```
pattern=ALARME_TRAP Host (\S+) Net (\S+) (\S+) estado INTERMITENTE
desc=alarme trap $1 $2 host $3 $5 estado instavel
action=write SEC NMSIS %t Host $1 Net $2 $3 MAJOR LIGAÇAO INSTAVEL
window=40
thresh=2
desc2=alarme_trap $1 $2 Host $3 $5 novamente estavel
action2=write SEC NMSIS %t Host $1 Net $2 $3 NORMAL ESTAVEL
window2=60
thresh2=0



# traps Cisco SW com correlação de intermitência e correlação com DB NMSIS
netbox.
# expressão regular:(\d+-\d+-
\d+)\s(\d+:\d+:\d+)\s((\d+\.\d+\.\d+\.)\d+).*enterprises.45.3.35.1\sLink\sDown\
sTrap\s.*INTEGER:\s(\d+)



type=PairWithWindow
ptype=RegExp
pattern=(\d+-\d+-
\d+)\s(\d+:\d+:\d+)\s((\d+\.\d+\.\d+\.)\d+).*enterprises.45.3.35.1\sLink\sDown\
sTrap\s.*INTEGER:\s(\d+)
desc=ALARME TRAP $1 $2 Host $3 Net $4 Port $5 estado DOWN
action=event %s;
continue2=TakeNext
ptype2=RegExp
pattern2=\s$3.*Link\sup\sTrap\s.*INTEGER:\s$5
desc2=ALARME TRAP Host %3 Net %4 %5 estado INTERMITENTE
action2=event %t %s ;
window=20

type=SingleWithSuppress
continue=TakeNext
ptype=RegExp
pattern=ALARME_TRAP (\S+) (\S+) Host (\S+) Net (\S+) Port (\S+) estado DOWN
desc=EVENTO_TRAP SUPRIME estado DOWN
action=reset +1 %s;
window=20


type=Pair
ptype=RegExp
pattern=ALARME_TRAP (\S+) (\S+) Host (\S+) Net (\S+) Port (\S+) estado DOWN
desc=alarme trap $1 $2 host $3 net $4 port $5 estado down
action=event %s; \
write SEC_NMSIS %t Host $3 Net $4 Port $5 MAJOR DOWN;
ptype2=RegExp
pattern2=\s$3.*STRING:\s$5\s.*STRING:\s\"up"
desc2=alarme_trap host %3 %5 state up
action2=write SEC NMSIS %t Host %3 Net %4 Port %5 NORMAL UP
window=86400


type=Pair
ptype=RegExp
pattern=alarme trap (\S+) (\S+) host (\S+) net (\S+) port (\S+) estado down
desc=EVENTO_TRAP Elemento $3 da rede $4 port $5 DOWN "VERIFICA EVENTO_NMSIS
[netbox]"
action=write SEC NMSIS %t %s
continue2=TakeNext
ptype2=RegExp
pattern2=.*ElementoID:\s(\d+)\sCategoria:\s(\w+)\sIP:\s(($4)\d+)\sSalaID:\s(\w+
)\sEstado:\sn
desc2=ALARME_NMSIS [netbox] Elemento: $1 Cat: $2 Sala: $5 " [$3] não acessivel
devido falha no elemento [%3] porto [%5] identificado"
action2=write SEC NMSIS %t %s
window=86400


type=SingleWith2Thresholds
ptype=RegExp
pattern=ALARME TRAP Host (\S+) Net (\S+) (\S+) estado INTERMITENTE
desc=alarme trap $1 $2 host $3 Port $5 estado instavel
action=write SEC_NMSIS %t Host $1 Net $2 Port $3 MAJOR LIGAÇAO INSTAVEL
```

```
window=40
thresh=2
desc2=alarme trap $1 $2 Host $3 $5 novamente estavel
action2=write SEC NMSIS %t Host $1 Net $2 Port $3 NORMAL ESTAVEL
window2=60
thresh2=0


# Correlação com DB NMSIS netbox e service.


type=Pair
ptype=RegExp
pattern=.*ElementoID:\s(\d+)\sCategoria:\s(\w+)\sIP:\s([\d\.]+)\sSalaID:\s(\w+)
\sEstado:\sn
desc=ALARME_NMSIS [netbox] ElementoID: $1 Cat: $2 IP: $3 SalaID: $4 "Elemento
de rede [$3] não acessivel "
action=write SEC NMSIS %t %s
continue2=TakeNext
ptype2=RegExp
pattern2=\s$1.*ServiçoID:\s(\d+)\sActivoTimeout:\s(\w+)\sServiceType:\s(\w+)\sE
stado:\sn
desc2=ALARME_NMSIS [service] ServiçoID: $1 ServiceType: $3 "Quebra serviço
devido falha no elemento [%3]"
action2=write SEC NMSIS %t %s
window=20
```

Anexo L. Configuração do Switch Nortel, manual do utilizador “Using the BayStack 450 10/100/1000 Series Switch”, com o capítulos do manual 1.12-1.17, 3.15-3.17, 3.59-3.61, ou páginas do manual 44-49, 175-177, 219-221.

Caracteristicas gerais do BayStack 450:

BayStack 450 switches provide wire-speed switching that allows high-performance, low-cost connections to full-duplex and half-duplex 10/100/1000 Mb/s Ethernet local area networks (LANs):

- High-speed forwarding rate: Up to 3 million packets per second (peak)
- Store-and-forward switch: Full-performance forwarding at full line speed, using a 2.56 Gigabit/second switch fabric
- Learning rate: 3 million addresses per second (peak)
- Address database size: 16,000 entries at line rate (32,000 entries without flooding)
- Fail-safe stacking: Provides uninterrupted connectivity for up to eight units, with up to 224 ports stacked together as one managed unit (requires one optional BayStack 400-ST1 Cascade Module kit per stacked unit. See your Nortel Networks sales representative for ordering information).
- Spanning Tree Protocol (STP): Complies with IEEE 802.1D standard. STP can be disabled on the entire switch or stack, or on a per-port basis.
- SNMP agent support
- High-speed Uplink/Expansion Module slot: Allows you to attach optional media dependent adapters (MDAs) that support a range of media types.

- Rate limiting: Adjustable broadcast or IP multicast packet-rate limits for control of broadcast and IP multicast storms.
- Console/Comm port: Allows you to configure and manage the switch locally or remotely.
- IGMP snooping
- IEEE 802.1p prioritizing
- Virtual local area networks (VLANs), supporting:
    - IEEE 802.1Q port-based VLANs; --Protocol-based VLANs

TELNET:

Support for up to four simultaneous TELNET sessions; Optional password protection; Login time-out; Failed-login guard; Inactivity time-out; Allowed source addresses; Event logging

IEEE 802.3u-compliant autonegotiation ports, with four modes:

10BASE-T half-duplex; 10BASE-T full-duplex; 100BASE-TX half-duplex; 100BASE-TX full-duplex

MultiLink Trunking, supporting:

Switch-to-switch trunks; Switch-to-server trunks

Port mirroring (conversation steering)

Port-based; MAC address-based

Remote monitoring (RMON), with four groups integrated:

Statistics; History; Alarms; Events

MIBs

The BayStack 450 switch supports an SNMP agent with industry-standard MIBs, as well as private MIB extensions, which ensures compatibility with existing network management tools. The BayStack 450 switch supports the MIB-II

(RFC 1213), the Bridge MIB (RFC 1493), and the RMON MIB (RFC 1757), which provide access to detailed management statistics.

The following MIBs are supported:

- EAPOL (IEEE 802.1X Port Access Control MIB)
- SNMPv2 (RFC 1907)
- Bridge MIB (RFC 1493)
- Ethernet MIB (RFC 1643)
- RMON MIB (RFC 1757)
- MIB-II (RFC 1213)
- Interface MIB (RFC 1573)

ATM Forum LAN Emulation Client MIB

Nortel Networks proprietary MIBs:

s5Chas MIB; s5Agent MIB; s5 Ethernet Multi-segment Topology MIB; s5 Switch BaySecure MIB; Rapid City MIB

SNMP Traps

The BayStack 450 switch supports an SNMP agent with industry-standard SNMPv1 traps, as well as private SNMPv1 trap extensions (Table 1-3). RFC 1215 (Industry Standard):

(Ver tabela da página do manual 49 “Using the BayStack 450 10/100/1000 Series Switch”).

Anexo M. Caracteristicas técnicas do Switch catalyst C2900 seriem".

Feature Description

- **Performance and Configuration**
    - Autosensing transmission on 10/100 ports
    - Autonegotiation of half- and full-duplex operation on 10/100 ports
    - Full-duplex operation on 100BaseFX ports
    - Two high-speed expansion slots supporting 10BaseT/100BaseTX, 100BaseFX and future gigabit, asynchronous transfer mode (ATM) and Inter-Switch Link (ISL) modules (Catalyst 2916M XL only)
    - Fast EtherChannel support for connections of up to 800 Mbps between switches and servers
    - Support for 2048 MAC addresses
    - IEEE 802.1d Spanning-Tree Protocol support
    - 4 Mb shared-memory architecture
    - Cisco Group Management Protocol (CGMP) to limit the flooding of multicast traffic to predefined ports
    - Port security to prevent unauthorized access to the network
    - Broadcast storm control to prevent performance degradation from broadcast storms
    - Embedded RMON (four groups)
- **Management**

- Embedded World Wide Web interface for most management tasks
- Cisco IOS command-line interface (CLI) via the console port or Telnet
- CiscoView device-management application

- **Redundancy**
  - Connection for optional Cisco 600W AC redundant power system (RPS) as a backup

Anexo N. Caracteristicas técnicas do router C2600: “ Release Notes for Cisco IOS Release 12.0 Software Feature Packs — Cisco 2600 Series”

Anexo O. Guia genérico, da cisco de como configurar traps em linha de comando CLI: “Cisco IOS SNMP Traps Supported and How to Configure Them Document ID: 13506”.

Histórico

- 2 de Novembro de 2010, Versão 1.0, mailto:1920550@isep.ipp.pt
- 5 de Novembro de 2010, Versão 1.0a, mailto:1920550@isep.ipp.pt
- 6 de Dezembro de 2010, Versão 1.0b, mailto:1920550@isep.ipp.pt

$Id:MEEC-TESE-SCEN.docx v1.0b Date: 5-11-2010$