View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Diposit Digital de Documents de la UAB

1336 Sistema de monitorització Open Source. Projecte d'Empresa per Computer Technology Catalunya.

Memòria del Projecte Fi de Carrera d'Enginyeria en Informàtica realitzat per Gonzalo Fernández Ibiza amb dni 47647792-A i dirigit per Ramon Grau Sala del departament del D.A.C.S.O (Departament d'Arquitectura de Computadors i Sistemes Operatius)

Bellaterra, 29 d' abril del 2009

Universitat Autònoma de Barcelona

etse
Escola Tècnica Superior d'Enginyeria

El sotasignat, ... RAMON CRAU SOLÉ .....

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en

I per tal que consti firma la present.

Signat: ..... RAMON CRAU SOLÉ .....

Bellaterra, ...de.....de 200.....

Índex

ANNEXES

INTRODUCCIÓ:

0.1 Objectiu/s del projecte:

L'objectiu del projecte 1336 Sistema de monitorització Open Source per l'empresa Computer Technology Catalunya, consisteix en l'estudi, simulació i implantació d'un conjunt d'aplicacions que permeten tenir un control sobre possibles problemes que puguin succeir a la nostra xarxa. Aquest projecte és la solució als problemes de detecció d'errors en el funcionament de les infraestructures de networking de les que disposen els nostres clients.

Actualment, podem trobar moltes eines de Software que ens permeten realitzar aquestes tasques amb bona solvència. La diferència entre el fet d'escollir Software lliure o Software de pagament radica, evidentment, en el pressupost que volem destinar per desenvolupar una eina de control professional així com el suport que podem obtenir de una comunitat d'usuaris i desenvolupadors. És per això que he volgut agafar el protocol més estès, el SNMP. A la gran majoria de les empreses ens trobem amb eines de Software força potents dels quals cal destacar: HP Open View, SolarWinds, Sitescope, Nagios, Cacti, etc.. Aquestes eines són força útils però la funcionalitat i el preu d'unes i altres varia molt.

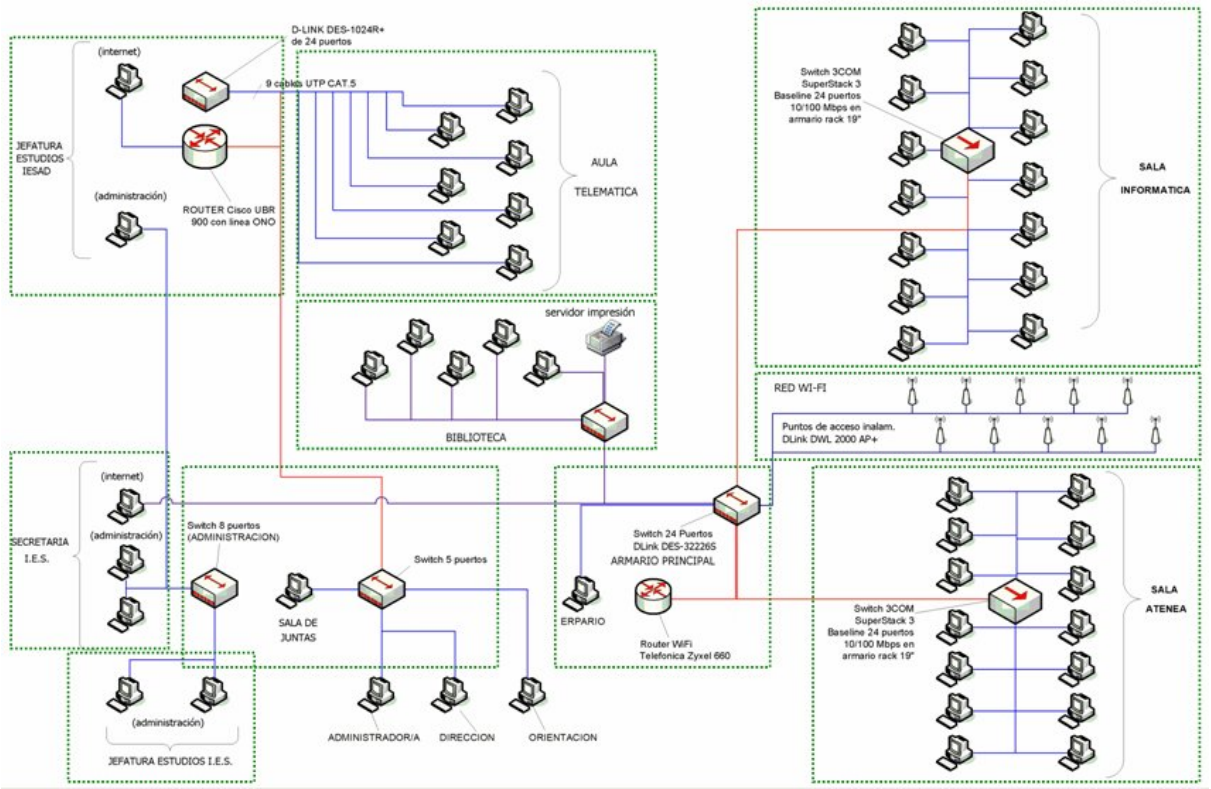
El fet de donar una solució fiable, flexible, estàndard i funcional als nostres clients, m'ha fet escollir una eina basada en Open Source per la possibilitat d' adaptar-la a les necessitats de cada client segons les seves especificacions. Al mateix temps, val a dir, que una eina de codi lliure ens proporciona un millor marge de negoci.

Avui en dia, gràcies a Internet i la gran quantitat d'informació que disposem online, podríem tenir instal·lat l'aplicació Nagios en qüestió d'hores. Hi ha molts manuals de com fer una instal·lació ràpida i guiada de la majoria de les eines de codi lliure. És per això que crec que el valor afegit que pot aportar un enginyer en aquest projecte és la possibilitat de definir uns patrons per tal de realitzar monitoritzacions avançades.

El control d'una xarxa és complex i requereix un gran coneixement del funcionament dels elements que hi han connectats. És important tenir clar què volem controlar i de quina manera. No es el mateix controlar si una màquina qualsevol està "viva" o si la temperatura del nostre servidor està superant els 29º Celsius ja que pel primer cas només caldria fer-li un ping i en el segon es necessiten eines addicionals i programació per tal d'obtenir aquesta informació del dispositiu.

Aquest projecte es centra en el disseny i implantació d'uns patrons per tal de poder controlar, mitjançant un protocol estàndard de monitorització, el màxim d' atributs possibles de cada servidor, switch, router, appliance, serveis, aplicacions, etc...connectats en una xarxa.

Aquí podem veure un exemple esquemàtic dels elements que s'acostumen a tenir connectats en una xarxa.



0.2 Introducció a l'estat de l'art del tema proposat

Què es Nagios?

Nagios és un sistema de control de xarxa. És capaç de controlar l'estat de Servidors, Pc's, switches, impressores, dispositius de xarxa, etc.., i serveis que s'especifiquin com podria ser l'estat d'una aplicació concreta o el servei d'un sistema operatiu.

En el moment en el que el comportament d'algun element controlat canvia d'estat i l'estat és una estat no desitjat o inesperat, ens avisa mitjançant una alarma, reportant que tenim un element amb problemes. D'aquesta manera, quant tenim un problema amb algun dels dispositius , no perdem temps en intentar esbrinar què està succeint a la nostra plataforma.

L'aplicació de Nagios permet realitzar plugins addicionals que permeten adaptar les comprovacions dels serveis segons les necessitats de cada situació. A més, el fet que sigui una eina de codi lliure fa que ens permeti constantment anar adaptant el codi a les nostres necessitats o a les necessitats dels nostres clients. Al mateix temps aquestes modificacions també les podrem trobar a varies comunitats d'usuaris que publiquen els seus desenvolupaments.

Les bases en les que fixaré el desenvolupament del projecte serà el protocol SNMP i el seu estàndard: RFC 1157 (SNMP, 1990) i RFC 3410 (SNMPv3, 2002).

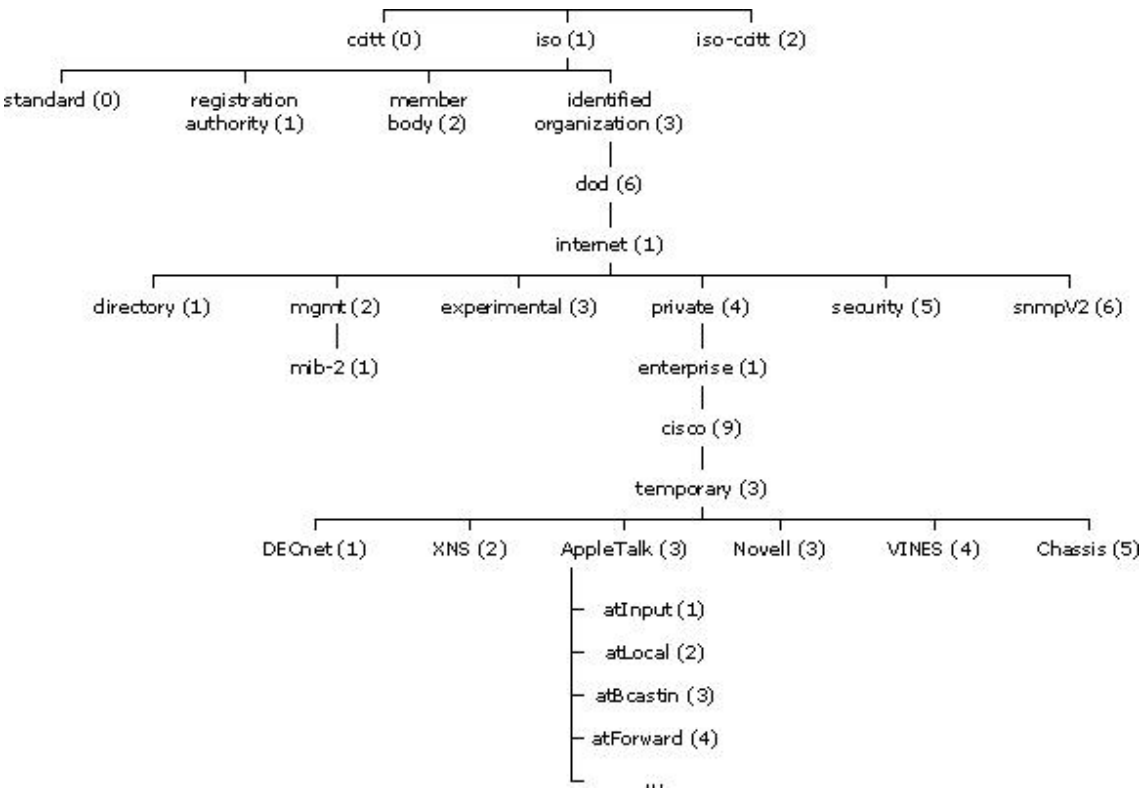
El protocol SNMP prové de les sigles (Simple Network Management Protocol). És un protocol de la capa d'aplicació del model TCP/IP i facilita l' intercanvi d' informació d'administració entre dispositius de xarxa. La funcionalitat principal del protocol snmp és la de proveir la possibilitat de detectar problemes. Aquesta funcionalitat és la que ens permet a nosaltres planificar el creixement de la xarxa.

Les versions de SNMP més utilitzades són dos, la versió 1 i la versió 2, encara que la versió 2 és, de llarg, la més utilitzada. Les diferències són poques. Bàsicament la versió aporta unes quantes millores a la versió 1. Al mateix temps s'ha de destacar que es va

crear una ultima versió l'any 2002, anomenada versió 3, que aportava considerables canvis però que encara no ha estat acceptada del tot.
Una monitorització basada en el protocol SNMP es basa en 3 components bàsics: dispositius a administrar, agents instal·lats als dispositius a controlar i els Sistemes administradors.

Els dispositius a administrar són nodes connectats en la mateixa xarxa i que acostumen a ser servidors, impressores, routers, swithces, etc.. Aquests nodes han de contenir un agent SNMP per tal de poder anar recopilant la informació necessària que en qualsevol moment pot ser consultada pels Sistemes administradors.

Els agents són mòduls de software que s'executen al sistema operatiu del node a administrar. La funcionalitat principal de l'agent consisteix en recopilar tota la informació necessària local del dispositiu a controlar. El coneixement d'aquest agent seria per exemple: l'estat de la memòria lliure, l'estat d'un disc dur, el tant per cent que s'està fent servir de cpu, etc... Tota aquesta informació es recull en una estructura jeràrquica com es mostra en la següent imatge:



El sistemes administradors són els sistemes que supervisen i controlen que els dispositius administrats estiguin funcionant correctament. Aquests estan constantment preguntant als nodes si algun atribut ha canviat d'estat.

Nagios®

Per què Nagios?

Els principals competidors del nostre producte Open Source són aplicacions de desenvolupadors privats per les quals s'han pagar costoses llicencies anualment. Els preus d'aquests productes privats acostumen a ser inaccessibles per a petites i mitjanes empreses ja que la seva infraestructura informàtica es compon de dos ó tres servidors.

Si intentem decidir a on volem enfocar el nostre producte i veiem que el mon empresarial al nostre país està per 90 % de PYMES, el que significa que la gran majoria de les corporacions són petites i mitjanes empreses, és encertat pensar que desenvolupar un producte que està dissenyat per aportar solucions a aquests tipus d'empreses, esdevé un producte que pot acaparar una gran quota de mercat.

Evidentment, les aplicacions amb el codi protegit recolzades amb grans pressupostos privats, simplifiquen les tasques d'instal·lació, configuració i manteniment i ens ofereixen grans possibilitats. Però hem de pensar que estem totalment restringits a l'hora de adaptar lo millor possible el software a les nostres necessitats ja que no podem modificar el codi de l'aplicació.

Això no significa que amb aquests productes no tindríem un bon control de la nostra xarxa, és més, segurament si explotéssim aquests productes trobaríem funcionalitats addicionals que no ens venen de sèrie amb Nagios. És per això que cal destacar que s'ha considerat en tot moment la necessitat de trobar la solució més econòmica i competent sense oblidar la necessitat d'estabilitat.

De totes maneres, cal repassar les tres aplicacions més utilitzades en aquest camp:

-HP Open View de Hewlett-Packard

-Orion de SolarWinds

-Sitescope de Mercury.

Hp Open View de Hewlett-Packard:

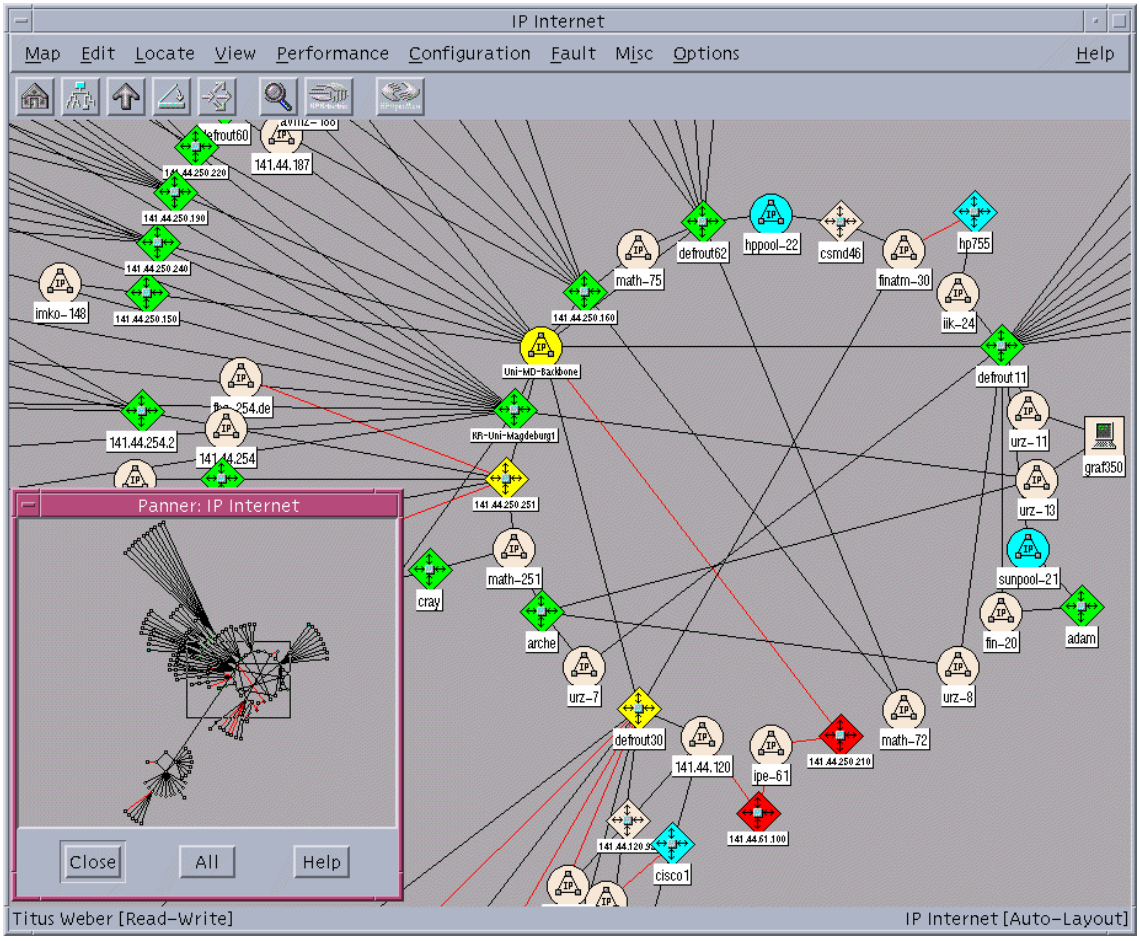
Actualment, aquest software de control de xarxes va ser pertany al fabricant informàtic Hewlett-Packard ja que al voltant del 2007 va ser reanomenat sota el Software de HP.

Si fem una descripció bàsica d'aquest producte, podríem dir que és una suite d'aplicacions de software que permeten controlar la gestió d'un sistema de xarxa a una organització de xarxa.

Aquestes aplicacions, d'una manera similar a Nagios, comproven l'estat dels dispositius que tenim donats d'alta a la nostra plataforma de control.

Hem de destacar que acostumen a fer servir el mateix disseny: dispositius a administrar, agents instal·lats als dispositius a controlar i els Sistemes administradors.

Les llicencies del software que hem de pagar per poder gaudir d'aquest producte ens fan arribar a la conclusió que únicament seria adequat per plataformes complexes.



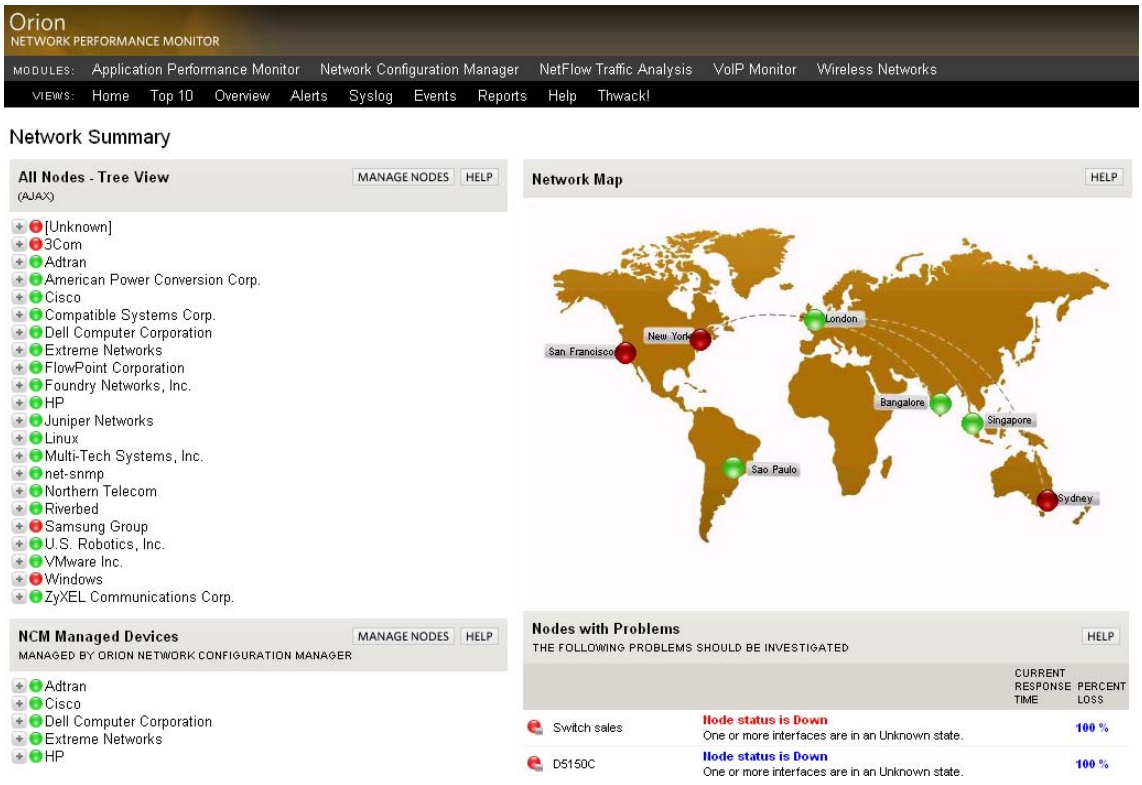
ORION de Solar Winds:

Solar Winds es corporació que ofereix molts productes diferents que ens poden proporcionar garanties de que la nostra plataforma informàtica disposarà quasi el 100% de disponibilitat dels recursos disponibles.

És una eina molt bona que ens permet visualitzar l'estat de la nostra xarxa d'una manera senzilla i pràctica que ens ajuda molt a interpretar els possibles problemes que podríem tenir en cas de caigudes dels punts crítics de la nostra xarxa.

Al igual que Hp Open View i Sitescope, aquesta eina és de pagament.

Personalment, es podria considerar apropiada per un perfil de corporació que tingui diferents sucursals arrel del mon i es necessiti centralitzar la monitorització a nivell global.



Sitescope de Mercury:

Al igual que Hp Open View, Sitescope de Mercury va ser adquirida recentment pel fabricant d'informàtica Hewlett-Packard.

La principal característica d'aquest software és que és una de les principals solucions que es basen en una filosofia no agressiva ja que no s'instal·la cap agent als dispositius a administrar. Simplement habilitant l'agent snmp en aquests dispositius ja podríem realitzar consultes al sistema operatiu sobre l'estat tant del software com del hardware.

Pel que fa a aquest aspecte, és molt similar a l'objectiu que volem arribar en aquest projecte ja que el seu disseny està centrat en consultes fent servir el protocol estàndard de SNMP. Al mateix temps cal destacar que la seva interfície es molt robusta i permet fer comprovacions dels dispositius en qüestió de segons. Per altre banda, aquest software es ven segons el numero de serveis a controlar. Encara i així no resulta econòmic per a petites i mitjanes empreses.

0.3 Motivació

El motiu que m'ha fet escollir aquest projecte ha estat el meu gran interès per les xarxes i la necessitat de realitzar una eina que em permeti donar un servei de qualitat als meus clients.
La majoria de les vegades, el problemes succeeixen quan no hi ha hagut una planificació de manteniment i en cap moment s'ha pensat en accions de prevenció per tal d'evitar que tinguem caigudes de serveis, màquines, servidors, etc... Donat què és molt difícil preveure amb exactitud quant es produirà un error, el que podem fer es realitzar un pla de contingència per tal que l'error tingui el menor impacte negatiu possible.
Encara i així, en el cas que vulguem determinar on està el problema per tal d'activar el pla de contingència, ens adonem que no és una tasca trivial, donat el gran numero de components i respectives funcionalitats que componen una xarxa.

Són moltes les empreses d'avui en dia que la seva productivitat depèn en gran part de les tecnologies de la informació i no es poden permetre perdre beneficis a causa de errors de la infraestructura informàtica. Va ser aquesta dependència a les tecnologies de la informació dels meus clients el que em va motivar a desenvolupar un producte enfocat a minimitzar el temps d'impacte dels errors i al mateix temps intentar prevenir-los.

Haig de reconèixer que el fet d'haver escollit una solució basada en codi lliure m'aporta una motivació extra i encara més si la instal·lació també es fa sobre un sistema operatiu basat també en codi lliure com és la distribució Ubuntu de Linux.

Crec fortament que el software de codi lliure tindrà un pes molt important al futur i encarà més si tenim en compte la crisi mundial que estem patint.

CAPÍTOL 1. Estudi de viabilitat del projecte

1.1. Anàlisi de requeriments

L'objectiu principal d'aquest projecte es basa en el disseny i implantació d'una solució destinada a donar una solució que permeti als nostres clients detectar en temps real qualsevol problema d'un Servei, Host ,etc...

L'elecció d'una aplicació basada en codi lliure implica que la viabilitat del projecte està a l'abast de qualsevol enginyer que s'impliqui en el desenvolupament de funcionalitats basats en un protocol estàndard i reconegut per una comunitat internacional.

Inicialment, la idea d'implementar un control de xarxa basat en codi lliure significa el fet de fer una anàlisi per cada cas concret sobre els diferents dispositius que volguéssim controlar.

És per això que cal fer un anàlisi de cada element hardware a controlar, cada aplicació que volem determinar l'estat o la seva funcionalitat, Appliances específics com podrien ser un Antispam, dispositius de xarxa com són el switches, etc..

Cada element a controlar té els seus propis atributs pels quals podem preguntar mitjançant el protocol snmp. La viabilitat d'implementar aquests controls passa per poder obtenir els valors d'aquests atributs i saber interpretar-los correctament. És aquí on entra el projecte.

Els requeriments per desenvolupar aquest projecte és un servidor o pc amb les eines instal·lades necessàries. És per això que les despeses per desenvolupar el projecte són quasi bé escasses. A més, el fet de realitzar la infraestructura amb una plataforma virtual el cost del servidor físic desapareix. En el cas d'haver de realitzar la plataforma per a un client concret, el cost real és petit comparat amb el que pot costar un servidor ja que Nagios no necessita un gran servidor per executar l'aplicació

1.2. Anàlisi de dispositius

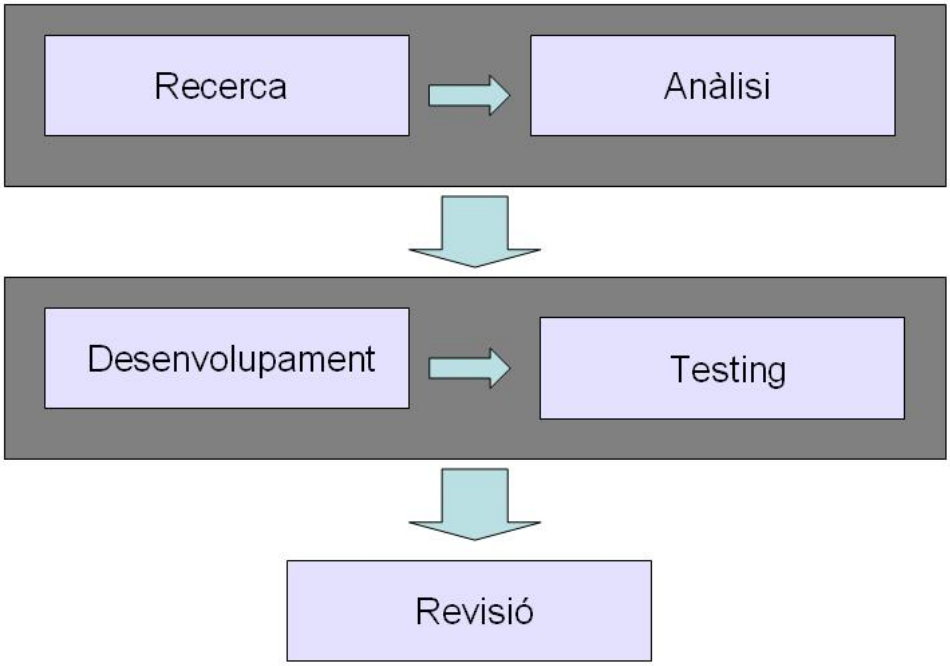
Els recursos necessaris pel bon funcionament del producte Nagios i el seus complements no són gaire elevats. Hem de pensar que tota el desenvolupament del producte Nagios ho fem des de la consola en mode text i en cap moment necessitem les X-windows del sistema operatiu. Això implica que qualsevol pc d'avui en dia supera els requeriments mínims per tal de tenir el funcionament esperat de l'aplicació. Concretament, únicament amb un pc amb un microprocessador a 1,5 Ghz, 512 MB de R.A.M, un disc dur de 15 GB ja seria suficient i una targeta de xarxa de 10 Mbps.

A l'actualitat, els pc més senzills ja es venen amb processadors que superen els 3 Ghz o microprocessadors amb dos i quatre nuclis amb velocitats que ronden els 2 GHz. Pel que fa a les R.A.M, tant a nivell de capacitat com de velocitat tenim moltes més prestacions a un preu econòmic de les que són bàsiques pel bon funcionament del aplicació. Si parlem de les tecnologies ethernet, veiem que la gran majoria de targetes ethernet ja funcionen amb velocitats de 1 Gbps. Donat que la tecnologia ethernet és la més estesa a nivell de xarxes de locals, ja que la majoria de pc's en porten una targeta, no s'ha considerat la necessitat de fer servir tecnologies de Fibra. Al mateix temps, aquestes no són més econòmiques ni aporten millores a les prestacions del producte.

El fet de no necessitar un servidor de grans prestacions fa més barat el pressupost final.
Les notificacions dels problemes que es pugessin detectar al sistema es poden reportar fent servir correu electrònic, Telèfon, WinPopup message, Yahoo, ICQ, Messenger i Alertes d'àudio.
Personalment m'he decidit per fer servir com a principal eines de notificació, el correu electrònic i Windows Popups. Únicament en el cas de les notificacions via SMS, caldria un dispositius USB lector de preu molt econòmic i una aplicació addicional anomenada Y.A.P (Yet Another Pager Software) que ens permetria enviar els missatges directament al telèfon. De totes maneres, crec que aquesta opció és aplicable en casos extrems i/o en corporacions amb departament d'explotació amb personal de guàrdia fora d'horari laboral.

1.3. Planificació temporal

La planificació temporal d'aquest projecte final de carrera ha seguit els següents punts: S'han definit 5 fases generals en la realització del projecte.



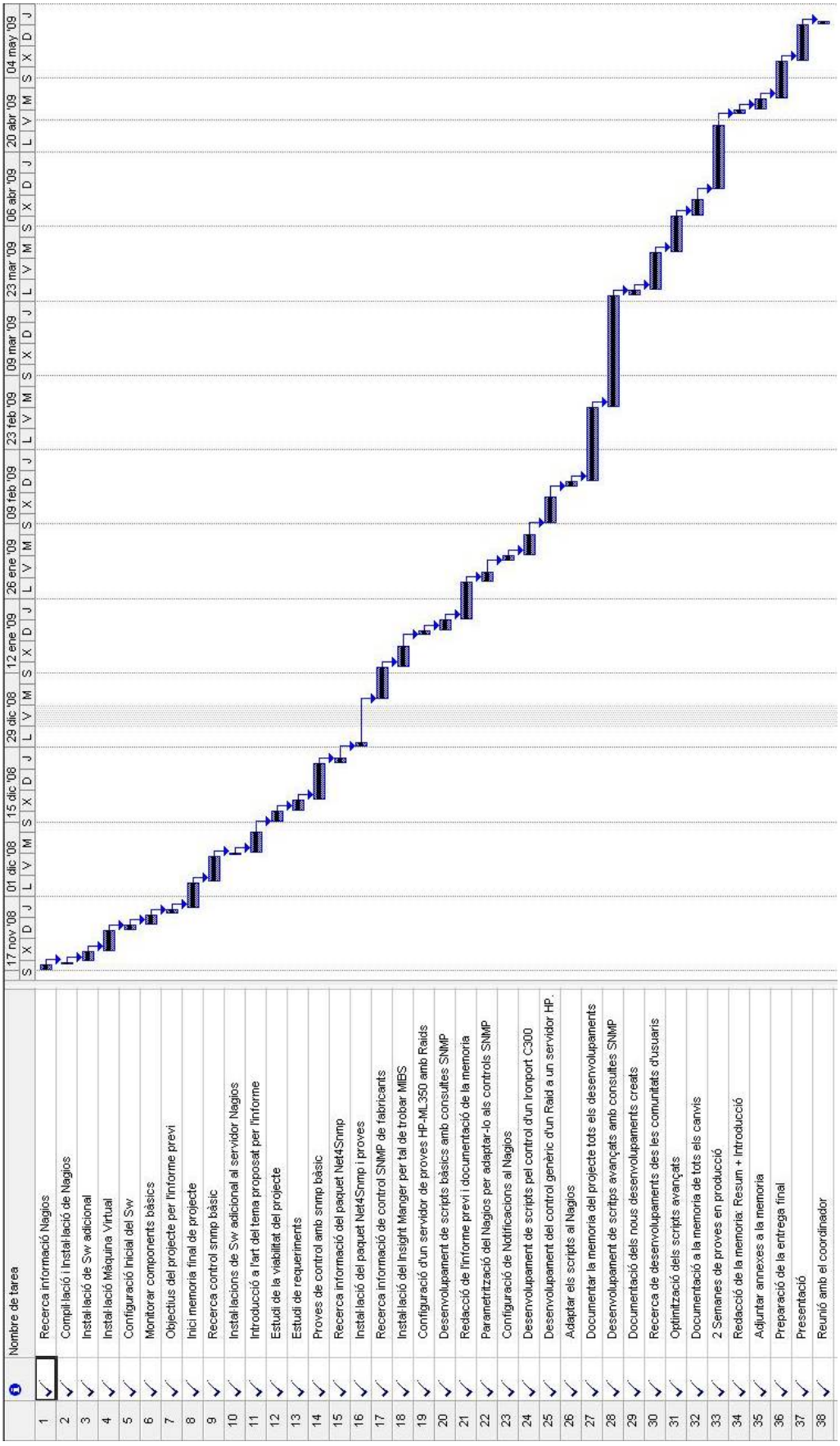
La dedicació en cada una de les cinc fases difereix considerablement degut a que el cost de realitzar el desenvolupament i comprovació és superior als costos de les altres fases. El fet de realitzar un disseny en la planificació del projecte en poques fases facilita el desenvolupament de la planificació de les tasques en cada una de les fases.

Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
Recerca de Informació	16 días	lun 17/11/08	lun 08/12/08	
Anàlisi	30 días	mar 09/12/08	lun 19/01/09	1
Desenvolpumant d'aplicacions	51 días	mar 20/01/09	mar 31/03/09	2
Testing	30 días	mié 01/04/09	mar 12/05/09	3
Revisió i Documentació	20 días	mié 13/05/09	mar 09/06/09	4

Les tasques individuals per tal d'assolir els objectius d'aquest projecte són les següents:

		Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	✓	Recerca informació Nagios	1 día	lun 17/11/08	lun 17/11/08	
2	✓	Compil·lació i Instal·lació de Nagios	4 horas	mar 18/11/08	mar 18/11/08	1
3	✓	Instal·lació de Sw adicional	16 horas	mar 18/11/08	jue 20/11/08	2
4	✓	Instal·lació Màquina Virtual	2 días	jue 20/11/08	lun 24/11/08	3
5	✓	Configuració Inicial del Sw	1 día	lun 24/11/08	mar 25/11/08	4
6	✓	Monitorar components bàsics	2 días	mar 25/11/08	jue 27/11/08	5
7	✓	Objectius del projecte per l'informe previ	1 día	jue 27/11/08	vie 28/11/08	6
8	✓	Inici memoria final de projecte	3 días	vie 28/11/08	mié 03/12/08	7
9	✓	Recerca control snmp bàsic	3 días	mié 03/12/08	lun 08/12/08	8
10	✓	Instal·lacions de Sw adicional al servidor Nagios	4 horas	lun 08/12/08	lun 08/12/08	9
11	✓	Introducció a l'art del tema proposat per l'informe	4 días	mar 09/12/08	vie 12/12/08	10
12	✓	Estudi de la viabilitat del projecte	2 días	lun 15/12/08	mar 16/12/08	11
13	✓	Estudi de requeriments	2 días	mié 17/12/08	jue 18/12/08	12
14	✓	Proves de control amb snmp bàsic	1 sem	vie 19/12/08	jue 25/12/08	13
15	✓	Recerca informació del paquet Net4Snmp	1 día	vie 26/12/08	vie 26/12/08	14
16	✓	Instal·lació del paquet Net4Snmp i proves	1 día	lun 29/12/08	lun 29/12/08	15
17	✓	Recerca informació de control SNMP de fabricants	4 días	mié 07/01/09	lun 12/01/09	16
18	✓	Instal·lació del Insight Manger per tal de trobar MIBS	4 días	mar 13/01/09	vie 16/01/09	17
19	✓	Configuració d'un servidor de proves HP-ML350 amb Raids	1 día	lun 19/01/09	lun 19/01/09	18
20	✓	Desenvolupament de scripts bàsics amb consultes SNMP	2 días	mar 20/01/09	mié 21/01/09	19
21	✓	Redacció de l'informe previ i documentació de la memoria	5 días	jue 22/01/09	mié 28/01/09	20
22	✓	Parametrització del Nagios per adaptar-lo als controls SNMP	2 días	jue 29/01/09	vie 30/01/09	21
23	✓	Configuració de Notificacions al Nagios	1 día	lun 02/02/09	lun 02/02/09	22
24	✓	Desenvolupament de scripts pel control d'un Ironport C300	4 días	mar 03/02/09	vie 06/02/09	23
25	✓	Desenvolupament del control genèric d'un Raid a un servidor HP.	1 sem	lun 09/02/09	vie 13/02/09	24
26	✓	Adaptar els scripts al Nagios	1 día	lun 16/02/09	lun 16/02/09	25
27	✓	Documentar la memoria del projecte tots els desenvolupaments	2 sem.	mar 17/02/09	lun 02/03/09	26
28	✓	Desenvolupament de scritps avançats amb consultes SNMP	3 sem.	mar 03/03/09	lun 23/03/09	27
29	✓	Documentació dels nous desenvolupaments creats	1 día	mar 24/03/09	mar 24/03/09	28
30	✓	Recerca de desenvolupaments des les comunitats d'usuaris	1 sem	mié 25/03/09	mar 31/03/09	29
31	✓	Optimització dels scripts avançats	1 sem	mié 01/04/09	mar 07/04/09	30
32	✓	Documentació a la memoria de tots els canvis	3 días	mié 08/04/09	vie 10/04/09	31
33	✓	2 Semanes de proves en producció	2 sem.	lun 13/04/09	vie 24/04/09	32
34	✓	Redacció de la memoria: Resum + Introducció	1 día	lun 27/04/09	lun 27/04/09	33
35	✓	Adjuntar annexes a la memoria	2 días	mar 28/04/09	mié 29/04/09	34
36	✓	Preparació de la entrega final	1 sem	jue 30/04/09	mié 06/05/09	35
37	✓	Presentació	1 sem	jue 07/05/09	mié 13/05/09	36
38	✓	Reunió amb el coordinador	4 horas	jue 14/05/09	jue 14/05/09	37

Gràfica de desenvolupament:



CAPÍTOL 2. Descripció de les eines utilitzades

2.1 Nagios

Com ja hem comentat a la introducció, Nagios és sistema de control de dispositius de xarxa i de les seves funcionalitats, basat en codi lliure i originalment dissenyat per efectuar-se sota Linux encara que es pot instal·lar en altes sistemes operatius orientats a les *X-Windows (interfície gràfica)* com podria ser la família Windows de Microsoft.

Les principals funcionalitats d'aquesta plataforma són:

- Control de serveis de la xarxa (SMTP, POP3, HTTP, NNTP, PING, etc.)
- Control dels recursos de les màquines (càrrega del processador, ús del disc, etc.)
- Disseny de plugins que permeten parametritzar l'aplicació.
- Execució de checks en paral·lel.
- Possibilitat de definir una jerarquia de hosts, serveis, etc...D'aquesta manera l'aplicació ens reporta l'error al node Pare.
- Permet fer notificacions via mail, Windows pop-up, sms, etc..
- Possibilitat de programar "disparadors" per tal que s'executin en cas de caiguda d'algun node o servei de manera proactiva.
- Rotació de logs automàtica.
- Suport per servidors redundants de monitorització.
- Interfície Web opcional per veure l'estat de la xarxa, notificacions i historial de problemes.

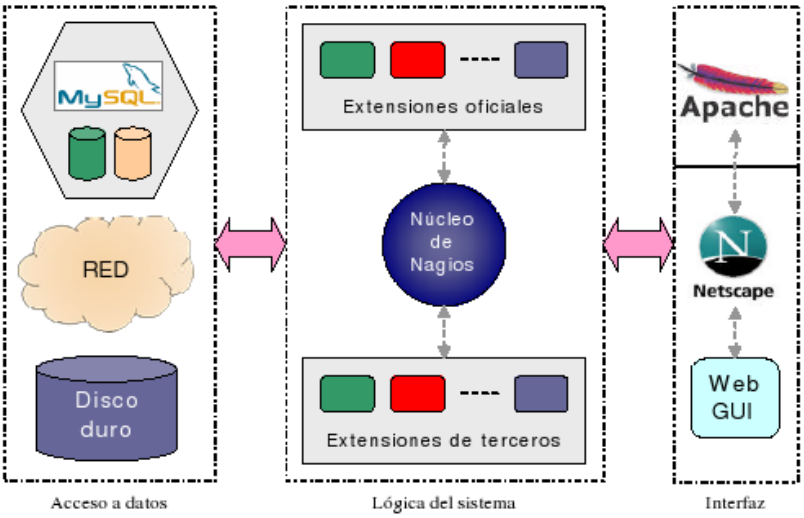
L'únic requeriment per poder executar Nagios amb bones condicions és una computadora amb Linux (o una variant de Unix) i un compilador C. Probablement necessitem tenir configurat TCP/IP, perquè les revisions dels serveis es realitzen mitjançant la xarxa.

Si volem fer servir les CGI's incloses a Nagios necessitarem el següent software a la nostra màquina Nagios:

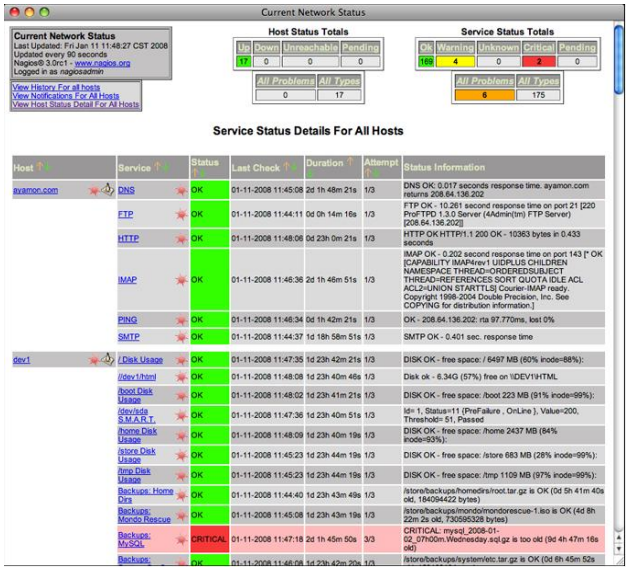
1.- Un servidors web (hem escollit Apache).

2. Thomas Boutell's biblioteca Gd versió1.6.3 o més gran (requerit pels CGIs de statusmap y trends)

L'estructura de Nagios es podria representar de la següent manera:



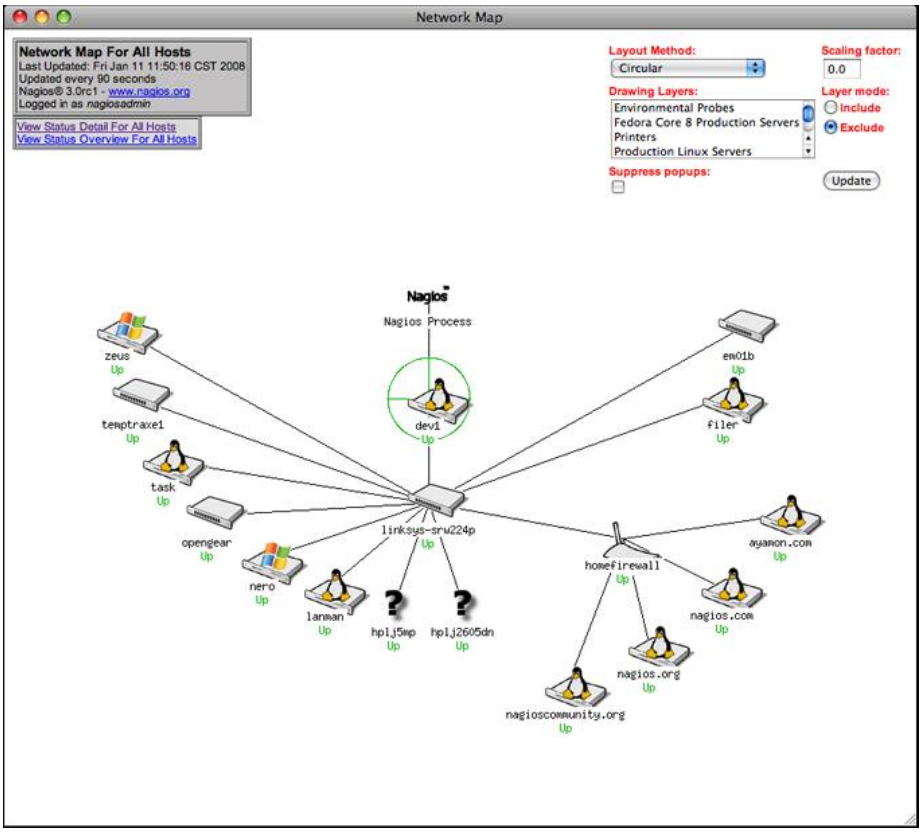
Un nucli de l'aplicació que forma la lògica de control de negoci de l'aplicació, que conté el software necessari per realitzar la monitorització dels serveis i màquines de la xarxa per les quals està preparat. Fa ús de diversos components que venen amb l'aplicació, i pot fer ús d'altres components realitzats per terceres persones. Encara que permeti la captura de paquets SNMP per notificar successos, Nagios no és un sistema de monitorització i gestió basat en la captura de "Traps". Bàsicament es fa servir gran quantitat de mòduls diferents com agents, scripts en snmp, etc... que permeten realitzar el control desitjat. Ens mostra els resultats de la monitorització i l'ús dels components a una interfície web mitjançant un conjunt de CGI's i pàgines web.



Aquesta interfície, permet a l'administrador una completa visió de que està succeint, en quina part de la nostra xarxa, i si estan ben programats els scripts o "checks", perquè. Per últim, si es compila amb paràmetres concrets, Nagios ens guardarà els històrics a una base de dades per si es dona el cas de parar el servei de Nagios, les dades no es perdin.

La interfície Web es simple i clara i de fàcil gestió. Una de les pàgines que ens aporten molta informació es la visualització dels serveis que tenim definits. D'aquesta maneta es pot visualitzar ràpidament quin serveis estan provocant problemes a la nostra infraestructura.

Una visualització també força utilitzada és la de a la nostra infraestructura és la de status Map. Gràcies a aquest tipus de visualització podem arribar a veure els nostres punts crítics i a saber interpretar i planificar el creixement de la nostra xarxa.



2.2 Snmp4Nagios.

Snmp4Nagio és una paquet de Nagios disponible als repositoris que ens permet fer el control dels monitors creats mitjançant consultes snmp contra el servidor monitoritzat.

És per això que cal fer primer una explicació mes profunda del protocol snmp i el més important pel nostre projecte, exposar amb detall què es una MIB i els seus OID's.

Base d'informació d'administració SNMP (MIB) i els OID's.
Una base d'informació d'administració MIB és una col·lecció d'informació que està organitzada jeràrquicament. Les MIB's son accedides fent servir el protocol d'administració de xarxa, com per exemple el SNMP.

Un objecte administrat té un únic valor associat a ell anomenat OID (Object ID). Aquest valor guarda una variable amb informació concreta. Podem diferenciar que hi ha dos tipus diferents d'objectes administrats: Escalars i tubulars. Els objectes escalars defineixen una simple instància d'un objecte mentre que els objectes tubulars defineixen múltiples instàncies d'un objecte relacionades que estan agrupades conjuntament en taules MIB.

Un exemple d'un objecte administrat és *atInput*, que és un objecte escalar que conté una simple instància d'un objecte, el valor sencer que indica el nombre total de paquets Apple Talk d'entrada sobre una interfície del router.

Pel que fa als OID's, únicament identifica un objecte administrat a la jerarquia MIB.

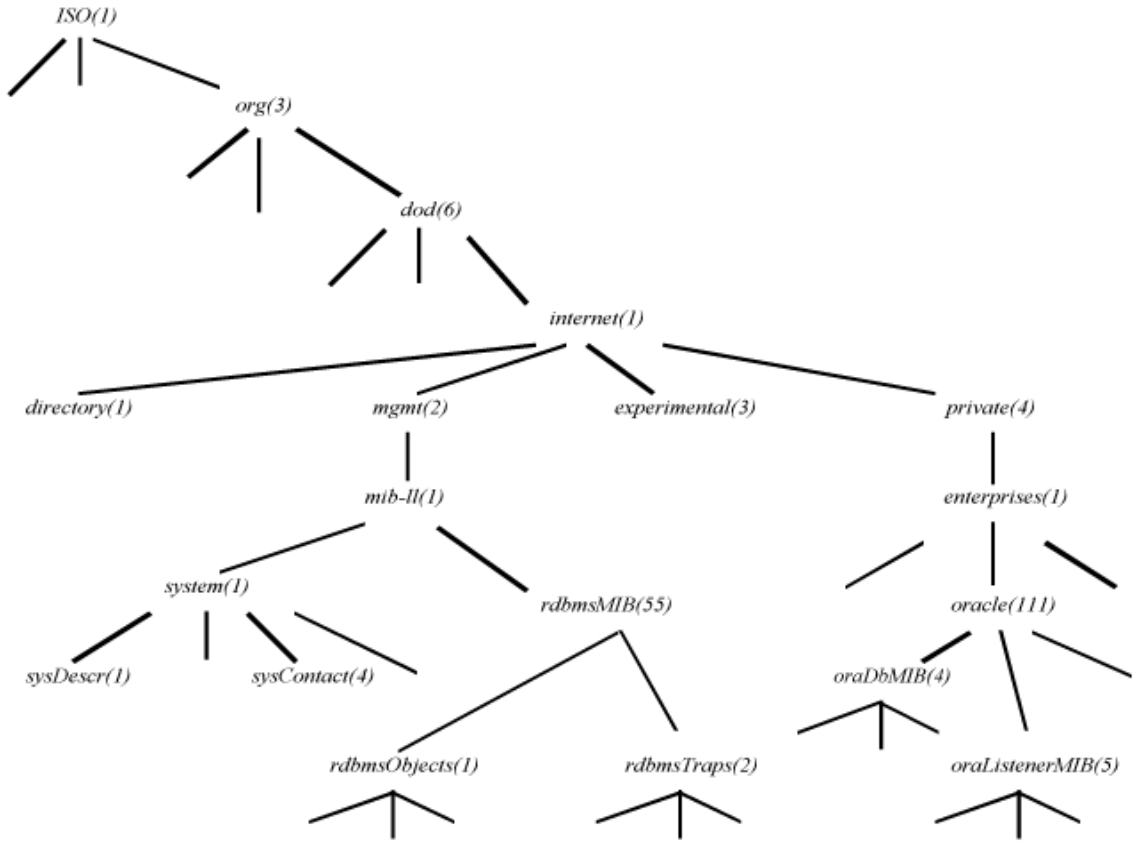
La jerarquia MIB pot ser representada com un arbre amb una arrel anònima i el seus nivells, que són assignats per diferents organitzacions.

Els identificadors dels objectes ubicats a la part superior de l'arbre pertanyen a diferents organitzacions estàndards, mentre que els identificadors dels objectes ubicats a la part inferior de l'arbre son col·locats per les organitzacions associades.

Els venedors de hardware poden definir branques privades que inclouen els objectes administrats pels seus productes. Les MIB's que no han estat estandarditzades típicament estan localitzades en la branca experimental.

L'objecte administrat atInput podria ser identificat pel nom de l'objecte iso.identified-organization.dod.internet.private.enterprise.cisco.temporary.AppleTalk.atInput o pel descriptor de l'objecte equivalent 1.3.6.1.4.1.9.3.3.1 .

L'arbre MIB genèric segueix el següent esquema:



Mentre la majoria dels plugins i checks que fa servir nagios fan servir MIBs estàndards o dependents d'agents, els checks que s'incorporen a la plataforma nagios amb aquest paquet, a diferència dels plugins per defecte, ens permet escanejar hardware concret dels fabricants com podria ser la temperatura interna del servidor, el nombre de revolucions per minut que està donant el ventilador d'un Appliance de la manera com defineix el fabricant del hardware. Al mateix temps ens permeten mantenir la possibilitat de controlar els logs i fer gràfiques de les dades recollides fent servir eines addicionals com seria les RRDTool de Tobias Oetiker.

El paquet SNMP4Nagios ens aporta la possibilitat de fer consultes amb el protocol snmp a dispositius de hardware fabricats per empreses com Brocade, Cisco, Compaq/HP, etc..

Les possibilitats que ens aporta aquest paquet les veiem amb els checks que incorpora aquest plugin. (Annex 1)*

2.3 Scripts

Per poder adaptar l'aplicació Nagios a les nostres necessitats és necessari adaptar el codi dels checks que ens proporciona o bé crear de nous per tal que l'aplicació els executi i ens retorni els valors que necessitem enregistrar.
El scripting en aquest projecte ha tingut un pes molt important ja que s'ha aprofitat el fet que la màquina on hem instal·lat Nagios basada en Linux, ens permet executar scripts des de la consola. Hem veig obligat a nombrar com a mínim tres tipus de llenguatges que s'han fet servir. Els tres llenguatges són: C-shell, bash, Perl. El que he fet servir més és el Bash degut a que és el llenguatge més potent i versàtil i amb el que m'he sentit més còmode degut a les possibilitats que m'oferia.
Anem a veure que diu la wikipèdia sobre aquests llenguatges:

- **C Shell** és un llenguatge de programació, conegut també com csh y que va ser desenvolupat por Bill Joy a la Universitat de Berkeley, Califòrnia. Se li considera més idoni per programadors que Pouar-ne Shell (chi), amés reflexa que la informàtica s'estava fent més interactiva. Avui en dia, l'original C Shell, no gaudeix d'un ampli ús en Unix; ha estat superat per altres Shells [1], como Tenex C Shell (tcsh) basat en el codi original C Shell, però agregant finalitzacions de noms a fitxers, y edició a comandes de línea, comparables a Korn Shell (Ksh), y el GNU Bourne-Again Shell (Bash).

- **Bash** és un shell de Unix (intèrpret d'ordres de Unix) escrit pel projecte GNU. El seu nom és un acrònim de bourne-again shell (una altre shell bourne) — fent un joc de paraules (born-again significa renaixement) sobre el Bourne shell (sh), que va ser un dels primers intèrprets importants de Unix. Cap al 1978 el intèrpret Bourne era el intèrpret distribuït amb Unix Version 7. Stephen Bourne,

investigador dels Laboratoris Bell, va escriure el intèrpret Bourne original . Brian Fox va escriure el intèrpret bash al 1987. Al 1990, Chet Ramey es va convertir en el seu principal desenvolupador. Bash es el intèrpret predeterminat a la majoria de sistemes GNU/Linux, amés de Mac OS X Tiger, y pot executar-se en la majoria de los sistemes operatius tipus Unix. També s'ha portat a Microsoft Windows pel projecte Cygwin.

- **Perl** és un llenguatge de programació dissenyat per Larry Wall creat al 1987. Perl pren les característiques del C, del llenguatge interpretat shell (sh), AWK, sed, Lisp y, en un grau inferior, de molts altres llenguatges de programació. Estructuralment, Perl està basat en un estil de blocs com els del C o AWK, y va ser àmpliament adaptat per la seva destresa en el processat de text y no tenir ninguna de las limitacions dels altres llenguatges de script.

CAPÍTOL 3. Instal·lació bàsica i configuració inicial

3.1 Instal·lació d'una màquina virtual

Encara que per l'aplicació Nagios no requereix gaires prestacions per poder ser executada i donat que a Comtech disposem d'eines de virtualització com Vmware Infraestructure, en contes de fer la instal·lació en un pc de sobretaula vaig preferir fer una instal·lació virtual.

Vaig agafar aquesta decisió ja que molts dels nostres clients disposen de plataformes virtuals Vmware el que em permet fer exportacions de les màquines Nagios que tingui creades a les nostres oficines, configurant-les prèviament i exportant-les amb una memòria portàtil tipus USB. 'ha creat una màquina virtual a l'empresa per tal de realitzar el projecte.

L'eina que he fet servir és el Vmware Infraestructure Client per tal de gestionar els servidors de màquines virtuals basats en servidors virtuals SX-servers.

La màquina virtual instal·lada porta un Sistema operatiu basat en codi lliure per tal de ser fidel a la filosofia inicial del projecte. És per això que vaig optar per basar-me en una distribució de Linux com és la distribució Ubuntu 8.0.4 .

Actualment, tenim varies distribucions i la gran majoria ens anirien igualment bé, però he de considerar que dintre dels pocs requeriments que tenim, haig d'agafar la distribució que més suport rebi de la comunitat d'usuaris i més facilitats ens aporti pel nostre desenvolupament. Tenint clars aquests punts, només cal donar una ullada per la quantitat de guies que tenim de la distribució Ubuntu de Linux. Segurament podrem trobar explicat amb detalls com podrien fer una instal·lació bàsica de qualsevol aplicació de codi lliure ja sigui als repositori i directament des de el codi, inclosa l'aplicació de Nagios.

3.2 Instal·lació Apache

Una vegada tenim una màquina amb el sistema operatiu desitjat instal·lat, ja podríem començar el procés de compilació i instal·lació. Les comandes que veurem a continuació han estat testades sota la distribució Ubuntu 8.0.4 amb resultats satisfactoris.

Primerament per poder completar tot el procés amb èxit haurem d'instal·lar software addicional necessari com servidor web, llibreries de compilació, etc...i seguir els següents passos.

1.- Instal·lem Apache, Build-essential i llibreries:

```
sudo apt-get install apache2
sudo apt-get install build-essential
sudo apt-get install libgd2-xpm-dev
```

2.-Creació de la informació de la conta: Creació d'un usuari "nagios" i li posem password.

```
sudo -s
/usr/sbin/useradd -m nagios
passwd nagios
```

3.- Creació del grup Nagios.

```
/usr/sbin/groupadd nagios
/usr/sbin/usermod -G nagios nagios

/usr/sbin/groupadd nagcmd
/usr/sbin/usermod -a -G nagcmd nagios
/usr/sbin/usermod -a -G nagcmd www-data
```

4.-Descàrrega del fitxers a compilar e instal·lar.

```
mkdir /opt/downloads/
cd /opt/donwloads
wget http://osdn.dl.sourceforge.net/sourceforge/nagios/nagios-
3.0.5.tar.gz
```

```
wget http://osdn.dl.sourceforge.net/sourceforge/nagiosplug/nagios-plugins-1.4.11.tar.gz
```

5.-Modifiquem el fitxer de configuració de l'apache.

```
ScriptAlias /nagios/cgibin/
/usr/local/nagios/sbin/
<Directory "/usr/local/nagios/sbin/">
AllowOverride AuthConfig
Options ExecCGI

Order allow,deny
Allow from all
</Directory>


Alias /nagios/ /usr/local/nagios/share/
<Directory "/usr/local/nagios/share">
Options None
AllowOverride AuthConfig
Order allow,deny
Allow from all
</Directory>
```

Una vegada fet aquest canvi reiniciem el servei del nostre servidor web Apache.

3.3 Compilació i instal·lació de Nagios 3.0.X

Ja tenim la nostra màquina preparada per fer la instal·lació principal de l'aplicació.
Hem descarregat el fitxers fonts i els plugins per poder començar. Hem veig obligat a comentar que aquest mètode d'instal·lació no és l'únic i podríem haver escollit vies més directes que ens permet aquesta distribució (Ubuntu) com és la comanda apt que t'instal·la paquets del repositori amb gran facilitat i sense compilacions. Però no vull que es condicionar el projecte i prefereixo deixar obertes altres vies amb aquesta guia.

Anem llavors a la compilació i instal·lació:

1.- Descomprimim els paquets descarregats, compilem i instal·lem.

```
tar xzf nagios-3.0.5.tar.gz
```

```
cd nagios-3.0.5
./configure --with-command-group=nagcmd
make all
make install
make install-init
make install-config
make install-commandmode
```

4.-Configurar el “web interface”. Creem l’usuari nagiosadmin que per defecte té permisos d’administrador al fitxer cgi.cfg

```
make install-webconf
htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
/etc/init.d/apache2 reload
```

5.- En cas que necessitem usuaris amb perfils de exclusius de visualització a l’eina, necessitarem un usuari operador.

```
htpasswd /usr/local/nagios/etc/htpasswd.users operador
```

6.- Hem de modificar el fitxer cgi.cfg per tal de donar permisos a l’usuari operador:

```
authorized_for_system_information=nagiosadmin,operador
authorized_for_configuration_information=nagiosadmin,operador
authorized_for_all_services=nagiosadmin,operador
authorized_for_all_hosts=nagiosadmin,operador
```

Ja el tenim creat. Seguim al directori /opt/downloads

7.- Anem a fer la compilació i instal·lació dels plugins bàsics.

```
tar xzf nagios-plugins-1.4.11.tar.gz
cd nagios-plugins-1.4.11
./configure --with-nagios-user=nagios --with-nagios-group=nagios
make
make install
ln -s /etc/init.d/nagios /etc/rcS.d/S99nagios
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
/etc/init.d/nagios start
```

8.- Una vegada arribat en aquest punt ja podem accedir a la interfície web.

http://ip_privada_del_servidor_nagios/nagios/

Usuari amb administració → usuari : nagiosadmin pwd: “el especificat en el punt 4”.
Usuari Operador (només lectura) → usuari: operador pwd “el especificat en el punt 5”.

3.4 Configuració bàsica de Nagios.

Ja podem tenir el nostre primer contacte amb l’aplicació. Si fem login a la interfície d’administració web ens adonarem que és molt pràctica i no gaire complicada.
Anem primer a intentar comprendre el conjunt de directoris i fitxers que pertanyen a l’aplicació i que la modificació d’aquests comportarà la personalització desitjada.

3.4.1 Els fitxers de configuració i els seus directoris

L’arrel del directori de l’aplicació quan fem una instal·lació per defecte el tenim a : /usr/local/nagios . Dintre del directori tenim el següent:

```
drwxrwsr-x 2 nagios nagios 4096 2008-05-21 15:39 bin
drwxrwsr-x 3 nagios nagios 4096 2008-11-05 16:22 etc
drwxrwsr-x 4 nagios nagios 4096 2008-11-20 16:10 libexec
drwxrwsr-x 2 nagios nagios 4096 2008-05-21 15:39 sbin
drwxrwsr-x 9 nagios nagios 4096 2008-05-21 17:44 share
drwxrwsr-x 5 nagios nagios 4096 2009-02-28 17:25 var
```

Ens hem de centrar en dos directoris que són els més importants: /usr/local/nagios/etc (fitxers de configuració) i /usr/local/nagios/libexec (plugins i checks artesanals).

/usr/local/nagios/etc

En aquest directori trobarem els fitxers de configuració de l’aplicació i els fitxers on definirem quins controls voldrem fer a la nostra xarxa.
Si fem un llistat d’aquest directori en s trobem amb els següents fitxers:

```
-rw-rw-r-- 1 nagios nagios 10458 2008-05-21 15:39 cgi.cfg
-rw-r--r-- 1 root   nagios    48 2008-06-10 16:12 htpasswd.users
-rw-rw-r-- 1 nagios nagios 42632 2008-08-01 08:59 nagios.cfg
drwxrwsr-x 2 nagios nagios  496 2009-02-23 08:59 objects(DIRECTORI)
```

```
-rw-rw---- 1 nagios nagios  1340 2008-05-21 15:39 resource.cfg
```

No profunditzarem en intentar explicar tots el paràmetres que podem modificar ja que, personalment, considero que no aporta cap valor afegit al fet de realitzar una explicació profunda de com parametritzar l'aplicació ja que en podem trobar un munt d'informació tant a la pàgina oficial de nagios com a altres pàgines de monitoritzacions relacionades amb els controls de xarxes. El que sí faré serà comentar cada fitxer quin rol té.

Cgi.cfg

El fitxer cgi.cfg, per exemple, és el principal fitxer de configuració on podem parametritzar l'aplicació nagios a la nostra conveniència.
El propi fitxer t'informa clarament amb comentaris sobre el significat de cada paràmetre. (*Annex 2*)*

Com podem veure en els comentaris tots els valors de les variables les podem modificar segons els nostres interessos.

Htpasswd.users

Pel que fa al fitxer `htpasswd.users` veurem que és aquí on es guarden els passwords encriptats dels usuaris de la nostra interfície web.

```
nagiosadmin:VBnAlGozpiatg  → Usuari d'administració
monitor:2f489JRkcmJb.      →  Usuari opeardor.
```

Nagios.cfg

Si donem una ullada al fitxer nagios.cfg veurem que és aquí on podrem determinar quins fitxers tindrà en compte la nostra aplicació en el moment que intenti executar-se. *(Annex 3)**

Són els cfg_files els que determinen els fitxers que acabem de veure els que nosaltres podrem modificar per crear hosts, serveis, etc... el fitxers que aquí no s'especifiquin, l'aplicació no els tindrà en compte.

Hem de recalcar que es podria dedicar un altre projecte final de carrera en fer un estudi de com millorar les parametritzacions en el fitxer nagios.cfg per tal de tenir el màxim rendiment. Concretament tenim unes 1200 línees de codi amb moltes variables que podem canviar. El que és cert és que els valors per defecte mai m'han donat cap problema i considero que són prou bons i no ens cal modificar res.

Resources.cfg

En aquest fitxer podrem definir macros generals a l'aplicació que podrem fer referència a elles amb els scripts. Bàsicament ens permet emmagatzemar variables globals.

Anirem doncs a veure els principals fitxers que defineixen els objectes que estem controlant i com. Recordaré que degut a que aquesta informació la podem trobar amb molta facilitat a Internet no m'esplaiaré gaire.

/usr/local/nagios/etc/objects

Dintre d'aquest directori es troba els fitxers pertanyen a un grup diferent dispositius del nostre sistema. En cada un dels fitxers seguirem el mateix patró per tal de definir els controls a realitzar:

Per fer-ho d'una manera ordenada i veient que les noves versions de nagios estan enfocant a la separació dels serveis en diferents fitxers segons als grups que pertanyen, he considerat necessari crear els següents fitxers de definició d'objectes:

```
-rw-rw-r-- 1 nagios nagios  8138 2008-11-20 15:54 commands.cfg
-rw-rw-r-- 1 nagios nagios  3960 2008-11-05 18:34 contacts.cfg
-rw-r--r-- 1 root   nagios  8502 2009-01-14 15:19 linux.cfg
-rw-rw-r-- 1 nagios nagios  5403 2008-05-21 15:39 localhost.cfg
-rw-r--r-- 1 root   nagios 33607 2008-11-20 16:11 network.cfg
-rw-rw-r-- 1 nagios nagios  3124 2008-05-21 15:39 printer.cfg
-rw-rw-r-- 1 nagios nagios  3293 2008-05-21 15:39 switch.cfg
-rw-rw-r-- 1 nagios nagios 10812 2008-05-21 15:39 templates.cfg
-rw-rw-r-- 1 nagios nagios  3209 2008-05-21 15:39 timeperiods.cfg
-rw-r--r-- 1 root   nagios  4280 2008-12-09 15:47 vmwareservers.cfg
-rw-r--r-- 1 root   nagios 21210 2009-02-23 08:59 windowsComtch.cfg
```

Hi ha tres fitxers que recalcarem i veurem algun exemple de definició del fitxer

Commands.cfg

```
# 'check_local_disk' command definition
define command{
        command_name    check_local_disk
        command_line    $USER1$/check_disk -w $ARG1$ -c $ARG2$ -p
$ARG3$
        }
```

Podem veure que no deixa de ser un simple script que rep paràmetres quan s'executa i crida al script del directori /usr/local/nagios/libexec passant-li els paràmetres.

Contacts.cfg

```
define contact{
        contact_name                    nagiosadmin             ;
Short name of user
        use                             generic-contact         ;
Inherit default values from generic-contact template (defined
above)
        alias                           Nagios Admin            ;
Full name of user
        service_notification_options    w,u,c,r
        host_notification_options       d,u,r
        email                           goo@comtch.es ;
        }
```

En aquest fitxer definim els administradors que podran rebre notificacions quan s'especifiqui.

Timeperiods.cfg

```
define timeperiod{
        timeperiod_name 24x7
        alias           24 Hours A Day, 7 Days A Week
        sunday          00:00-24:00
        monday          00:00-24:00
        tuesday         00:00-24:00
        wednesday       00:00-24:00
        thursday        00:00-24:00
```

```
        friday          00:00-24:00
        saturday        00:00-24:00
        }
```

Queda clar que estem definir un període de notificació que es 24x7.

Pel que fa als altres fitxers i tal i com hem comentat abans, cada fitxer s'encarrega d'un grup definit de màquines. Personalment he optat per fer una classificació segons el sistema operatiu si estem parlant de servidors (Windows- Linux) i una classificació de funcionalitats en dispositius (impressores, switches, vmware, etc...)
En aquests fitxers definirem els objectes amb paràmetres com els següents:

→ Descripció del servei: service_description
→ Servidor al que pertany el servei: host_name
→ Període de temps que s'ha de fer el control: check_period
→ Comanda a aplicar per realitzar el control: check_command
→ Grup de contacte: contact_groups
→ Opcions de les notificacions: notifications_options

El fitxer templates.cfg ens mostra com definir cada paràmetre als nostres arxius de configuració:
Exemple de definició de host:

```
define host{
        name                            generic-host    ;
notifications_enabled           1               ;
event_handler_enabled           1               ;
flap_detection_enabled          1               ;
failure_prediction_enabled      1               ;
process_perf_data               1               ;
retain_status_information       1               ;
retain_nonstatus_information    1               ;
notification_period             24x7            ;
register                        0               ;
}

define service{
```

```
        name                            generic-service         ;
active_checks_enabled           1                       ;
passive_checks_enabled          1                       ;
parallelize_check               1                       ;
obsess_over_service             1                       ;
check_freshness                 0                       ;
notifications_enabled           1                       ;
event_handler_enabled           1                       ;
flap_detection_enabled          1                       ;
failure_prediction_enabled      1                       ;
process_perf_data               1                       ;
retain_status_information       1                       ;
retain_nonstatus_information    1                       ;
is_volatile                     0                       ;
check_period                    24x7            ;
max_check_attempts              3               ;
normal_check_interval           10              ;
contact_groups                  admins          ;
notification_options            w,u,c,r         ;
notification_period             24x7            ;
register                        0               ;
 }
```

/usr/local/nagios/libexec

Anem a centrar-nos en el directori /usr/local/nagios/libexec on trobarem tots el plugins per defecte i els plugins que hem instal·lat amb el paquet nagios-plugins.

Els plugins són scripts que s'encarreguen de fer la consulta snmp al host en qüestió preguntant-li sobre un servei determinat.

Aquest plugin, una vegada ha fet la pregunta, retorna per la sortida estàndard un string que es guardarà a la variable $OUTPUT$ i un valor numèric que s'emmagatzemarà a $HOSTSTATE$ o bé a $SERVICESTATE$.

Els tipus de retorns que podem tenir dels checks són:

-1 → Unknkow

0 → OK

1 → Warning

2 → Critical

Ara comentarem les possibilitats que ens ofereixen aquests scripts i per què els podem fer servir.
Nota: Aquests checks es troben al *(Annex 0)**

3.4.2 Notificacions

Anirem a comentar breument com configurar les notificacions ja que les considero de gran importància pels nostres objectius, principalment per que ens permetrà no haver d'estar constantment controlant la interfície web per veure si tenim algun problema a la nostra infraestructura a casa del client i només restar pendent de que la seva aplicació nagios ens informi sobre qualsevol problema que pugui aparèixer.

Per poder configurar les notificacions a la nostra aplicació haurem de modificar els següents fitxers:
1.- /usr/local/nagios/etc/objects/contacts.cfg i definir al servei en qüestió a quin grup o usuari volem que es notifiqui el problema.0
2.- /usr/local/nagios/etc/objects/fitxer_del_servei.cfg És aquí on tenim definit el servei que s'està controlant. Només caldrà incorporar un paràmetre per tal que es realitzi la notificació.

Anem a veure algun exemple:

Tal i com hem comentat hem de crear el contacte i el grup de contacte:

Editem llavors el fitxer /usr/local/nagios/etc/objects/contacts.cfg: (*Annex 4*)*

Com podem veure al fitxer contacts.cfg hem definit 4 usuaris: (`nagiosadmin, nagiosadmin2, nagiosadmin3,nagiosadmin4`) cada usuari es podria ser un administrador diferent i segons es agradi podem notificar el problema a qui vulguem dels usuaris que hem definit al fitxer.
Al mateix temps hem definit un grup de contactes que ens permet agrupar a diferents contactes amb diferents rols.

Això ens permet, per exemple, fer distinció entres els rols dels nostres usuaris de la xarxa. Un cas típic és definir usuaris “desenvolupadors”, administradors, director de projectes, etc... D’aquesta manera si considerem necessari podem avisar a un conjunt de persones només fent referència a un grup de contactes.

Ara només queda definir al servei que volem que en cas de caiguda, nagios ens notifiqui. *(Annex 5)**

Tal i com hem definit els serveis, en cas de que la memòria o la partició de la c: tingui algun problema al servidor SRV-SQL-01, ens reportarà via mail a tot el grup d’administradors qualsevol dels dos problemes.

Postfix

Totes aquestes configuracions s’han de fer a fitxers de l’aplicació Nagios. Però perquè realment ens arribin aquestes notificacions hem de tenir configurat algun servidor de correu a la pròpia màquina nagios o bé fer un relay del correu que generi nagios i enviar-ho al servidor de correu corporatiu.
En el meu cas i donat que disposem d’un servidor de correu corporatiu i optat per fer un relay de tots el mails que generava nagios i enviar-los al servidor Exchange corporatiu. Per poder fer això s’ha d’instal·lar el postfix amb la simple comanda

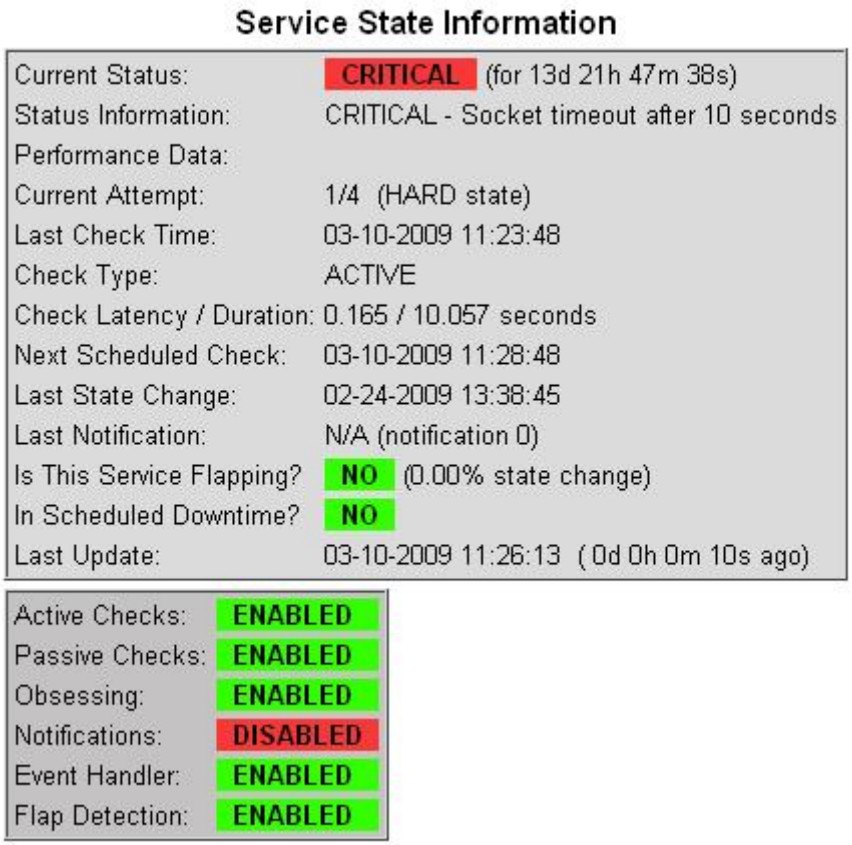
```
Apt-get install postfix
```

Una vegada el tenim instal·lat només queda editar el fitxer /etc/postfix/main.cf i deixar-ho de la següent manera: *(Annex 6)**

Ara ja podem fer les probes pertinents per verificar que les notificacions ens funcionen correctament.
Per fer probes només cal deshabilitar un servei des de windows i esperar a veure si rebem la notificació.
Abans de enviar el primer mail ha de tenir tants intents amb errors com el paràmetre: `max_check_attempts 4`

Com podem veure a la imatge, el Current Attempt defineix les vegades que s'ha provat de comprovar l'estat del servei. Quant el check ha superat el numero de vegades d'aquest paràmetre i no ha aconseguit obtenir un Ok, llavors notifica al contacte l'estat del problema.

En aquesta cas, i com es pot apreciar a la imatge es pot deshabilitar les notificacions si sabem que és un problema conegut.



3.4.3 Perfils d'autenticació Nagios.

El fet de realitzar una instal·lació d'aquesta infraestructura a les oficines del client, s'ha de tenir molt clar quins rols tindrà el client final. És per això que s'han creat tres tipus de contractes diferents per tal d'adaptar l'aplicació a les necessitats i coneixements del nostres clients. Cal remarcar, doncs, el rol que vol desenvolupar el nostres client amb l'aplicació i fins on vol arribar.

SERVEI D' IMPLANTACIÓ.

COMtech proporciona un servei d'implantació de la eina **Nagios**. Aquest servei s'ofereix amb 3 opcions, segons el nivell de gestió de l'aplicació que es desitgi realitzar:

• **Nivell administrador**. Instal·lació completa i formació a nivell d'administrador.
Per aquells clients que desitgin una completa implantació de l'aplicació, en la que ells mateixos s'encarreguin de la gestió, administració i manteniment.

• **Nivell operador avançat**. Instal·lació completa i formació a nivell d'usuari avançat. El client podrà realitzar una gestió parcial de l'aplicació (administració a nivell de CGIs, es a dir, deshabilitar alarmes, forçar comprovació de checks, etc.), però no incorporar serveis ni hosts a monitoritzar. En aquest nivell, el manteniment i l'administració de la mateixa la farà COMtech.

• **Nivell operador**. Instal·lació completa i formació bàsica a nivell d'usuari. El client utilitzarà l'aplicació a nivell d'usuari, però no podrà gestionar-la. La gestió completa de la eina la farà COMtech. Per aquells clients que volen conèixer l'estat de la seva xarxa per la detecció dels problemes i no haver de preocupar-se del manteniment de la mateixa.

Podem veure els tres possibles rols o perfils que poden desenvolupar els nostres clients.
Personalment, recomano deixar la gestió al nostre personal qualificat en contes de deixar-ho en mans del client. Bàsicament s'ha fet d'aquesta manera i no d'una altre per la senzilla raó que tenim clients amb el seu propi departament de sistemes que tenen els coneixements bàsics per poder gestionar i mantenir l'aplicació amb garanties d'èxit.

Una vegada tenim identificats quins rols volem aplicar als usuaris a la interfície web, anem doncs a modificar els fitxers de configuració per adaptar-los a les nostres necessitats.

Per poder crear usuaris a la nostra interfície web, hem d'executar la següent comanda d'Apache per tal de crear tants usuaris com vulguem crear.
La comanda en qüestió és la mateixa que fem servir per crear l'usuari operador:

```
htpasswd /usr/local/nagios/etc/htpasswd.users USUARI_A_CREAR
```

El que fem amb aquesta comanda és crear un usuari per poder accedir a la web amb autenticació. Una vegada tenim els usuaris creats, el que ens queda fer és determinar els rols i perfils de cada usuari. Per fer això, haurem d'editar els fitxers de configuració de Nagios per tal determinar els permisos de cada usuari.

```
vi /usr/local/nagios/etc/cgi.cfg
```

Ens ha d'aparèixer un fitxer amb el següent contingut. Posarem en negreta on hem d'incloure el text.

Justament on podem veure el nom nagiosadmin, podem substituir o incloure usuaris nous amb diferents rols d'administració.

Anem doncs a donar un cop d'ull els diferents perfils:

 # SYSTEM/PROCESS INFORMATION ACCESS
Aquesta opció dona accés a veure la informació dels processos Nagios que ens aporta el "Extended Information CGI" (extinfo.cgi). Per defecte ningú té accés fins que no s'especifica que no es vol autorització.

```
authorized_for_system_information=nagiosadmin
```

 # CONFIGURATION INFORMATION ACCESS
Aquest paràmetre ens permet veure tota la configuració que hi ha: hosts, comandes, etc...
En aquest cas els usuaris només podrien veure els hosts, serveis dels quals els usuaris apareixen com a contactes. Es pot fer servir un asterisk per determinar que tothom que s'hagi autenticat a la interfície web ho pugui veure.

authorized_for_configuration_information=**nagiosadmin**

 # SYSTEM/PROCESS COMMAND ACCESS
En aquest cas, els usuaris, si estan llistats en aquest paràmetre poden reiniciar o apagar comandes de Nagios via les comandes CGI. També poden activar o desactivar checks. Amb un asterisk, tothom autenticat pot estar autoritzat.

authorized_for_system_commands=**nagiosadmin**

 # GLOBAL HOST/SERVICE VIEW ACCESS
El global host/service view access ens dona autorització per visualitzar per tots el hosts/services que s'estan controlant. Per defecte només els usuaris que estan en les notificacions podrien veure l'estat. L'asterisc en aquest cas ens permet autenticar tothom.

authorized_for_all_services=**nagiosadmin**
authorized_for_all_hosts=**nagiosadmin**

 # GLOBAL HOST/SERVICE COMMAND ACCESS
El command access és un dels paràmetres que ens permet tots els cgi commands de la interfície web. D'aquesta manera tindrem accés total a l'aplicació.

authorized_for_all_service_commands=**nagiosadmin**
authorized_for_all_host_commands=**nagiosadmin**

CAPÍTOL 4. Configuració avançada

4.1 Instal·lació de Net4Snmp

Una vegada hem repassat les principals característiques que ens aporta nagios en la seva configuració per defecte, centrarem els nostres esforços en intentar descobrir les possibilitats que ens pot oferir treballar amb una eina de codi lliure on podem instal·lar paquets addicionals, plugins i adaptar-los a les nostres necessitats segons els nostres dissenys.

Amb la instal·lació per defecte de Nagios podem arribar a tenir la gran majoria de control de la nostre xarxa amb els components i plugins que vénen per defecte. L'objectiu d'aquest projecte era aportar un valor afegit que ens permetés tenir un control a més baix nivell que el propi sistema operatiu i per poder fer això és necessari instal·lar més components i paquets addicionals que ens permetin tenir aquest control a un nivell de complexitat superior. Això serà possible gràcies a un paquet que hem introduït al capítol 2.2.

Al mateix temps tindrem la necessitat d'incorporar agents snmp més potents que estan dissenyats pels propis fabricants de hardware i que seran els agents que ens respondran a les nostres consultes snmp ja que faran una extensió de la informació que pot disposar l'agent del sistema operatiu sobre els components dels quals té coneixement.

El Net-SNMP es una suite d'aplicacions que es fan servir per implementar snmpv1, snmpv2, snmpv3 sota ipv4i ipv6. Aquest paquet ens permet:

- Aplicacions de línea de comandes per:
  - Obtenir informació d'un dispositiu que escolti comandes snmp
  - Manipular configuració de la informació d'un dispositiu capaç d'entendre el protocol snmp (snmpset)
  - Obtenir col·leccions d'informació d'un dispositiu amb les comandes (snmpdf, snmpnetstat, snmpstatus).
  - Convertir formularis entre formats numèrics i de text de MIB OIDs, i mostrar la estructura del contingut de la MIB (snmptranslate).
- Una eina gràfica per visualitzar MIB's (tkmib).
- Un "daemon" per poder capturar els "traps" snmp (snmptrapd). Al mateix temps, les notificacions seleccionades es poden enregistrar a diferents sistemes de login.
- Un agent extensible per poder realitzar preguntes SNMP (snmpd).

- Una llibreria per poder desenvolupar noves aplicacions.

Una vegada hem donat un repàs a que consistia el paquet Net4Snmp, anem doncs a entrar a la part tècnica i anem a explicar quins són els passos que s'han de donar per realitzar la instal·lació d'aquests components.

El primer pas a realitzar és descarregar-nos el paquet snmp des de la web.

Per fer-ho d'una manera més ràpida he fet servir la comanda wget i el link directe on ho volia descarregar:

```
mkdir /opt/downloads

cd /opt/downloads

wget                http://downloads.sourceforge.net/net-snmp/net-snmp-5.4.2.1.tar.gz?use_mirror=fastbull
```

Ara procedim a fer la descompressió:

```
gzip –d net-snmp-5.4.2.1.tar.gz
```

I ara fem una extracció del tar:

```
tar –xvf net-snmp-5.4.2.1.tar
```

Compilació:

```
./configure
```

Al final de la compilació ens farà unes preguntes senzilles. Respondrem de la següent manera:

```
Default version of SNMP to use (3): 2

System Contact Information (root@): X@Y.es

System Location (Unknown): Barcelona

Location to write logfile (/var/log/snmpd.log): (ho deixem per defecte)

Location to write persistent information (/var/net-snmp): (ho deixem per defecte)
```

Fem el make

```
make
```

I fem el make install

```
make install
```

Aquests passos poden variar segons el tipus de sistema operatiu i hardware en el que s'estigui fent la instal·lació. És per això, que val a dir que aquesta instal·lació s'ha realitzat sota una distribució de Linux basat en Ubuntu 8.0.4.

El nostre objectiu principal no és el de fer servir els nous scripts que incorpora el nou paquet sinó el fet de tenir les llibreries i eines necessàries per fer els nostres propis scripts.

4.2 Disseny de scripts basats en SNMP.

La complexitat de l'objectiu de controlar l'estat concret d'un paràmetre d'un dispositiu qualsevol és directament proporcional a la complexitat de saber quin OID guarda el valor concret de l'estat que nosaltres volem saber.

Realment, la tasca d'intentar associar un OID a un atribut conegut, com per exemple: numero de processos en execució, no és fàcil ja que moltes vegades haurem de consultar a Internet o bé als propis fabricants en cas del hardware. És per això que la millor manera de planificar els monitors a controlar és centralitzar-me en un dispositiu concret i realitzar un *snmpwalk* per veure quina informació ens proporciona l'agent snmp del dispositiu. Una vegada tenim la informació hem d'interpretar-la per tal de detectar el paràmetre que ens interessa.

El primer objectiu que em vaig proposar va ser el d'intentar controlar un Appliance Antispam del fabricant Cisco Systems. El dispositiu en qüestió és un Ironport C300.
D'aquesta menar podria saber l'estat d'un dels dispositius amb més productivitat de l'empresa i al mateix temps tant crític degut al servei de *relay* que proporcionem als nostres clients.

Una vegada tenim clar quin dispositiu volem afegir a la nostra aplicació Nagios, anem a seguir el següents passos:

1.- Habilitar al dispositiu a controlar les consultes snmp des de Nagios.

2.- Realitzar un snmpwalk i exportar-ho a un fitxer.

3.- Analitzar i detectar els atributs que volem controlar (OID).

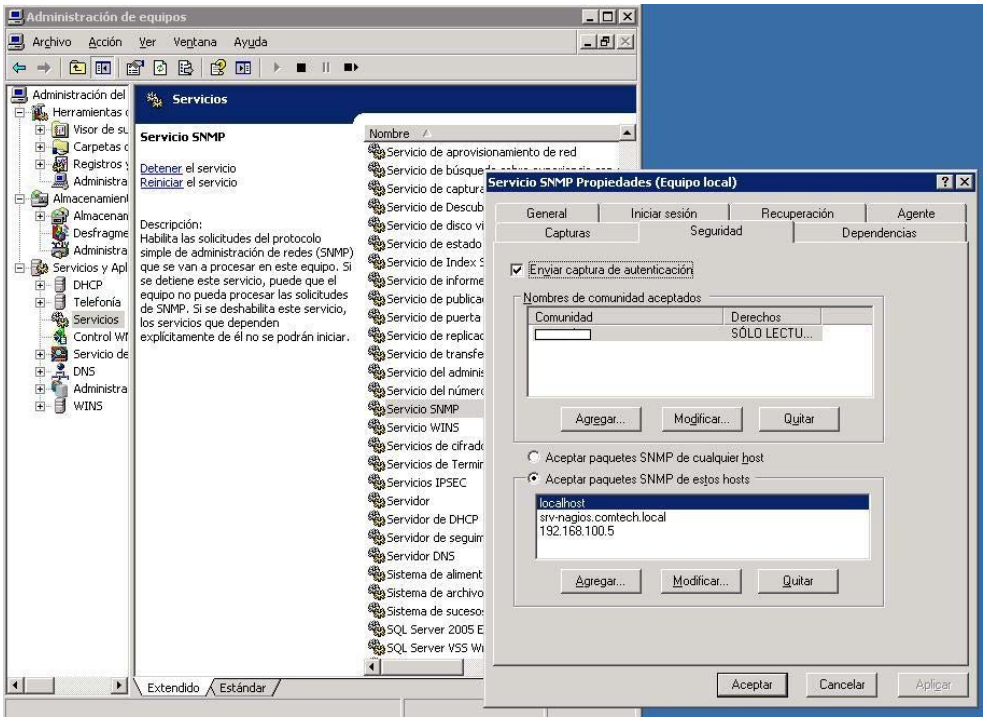
4.- Realitzar un script en bash

5.- Modificar l'aplicació Nagios per que tingui en compte els nous scripts.

Donat que els dispositiu que es poden arribar a controlar són tant diferents uns dels altres posarem els exemples més comuns per poder donar una idea clara de com fer-ho per qualsevol altre dispositiu.

1.- Habilitar al dispositiu a controlar les consultes snmp des de Nagios

La nostra primera fita serà habilitar a un servidor qualsevol les consultes del servidor Nagios a l'agent snmp. En el cas dels servidors Windows, per exemple, és tan senzill com editar les propietats del servei snmp del sistema operatiu i incloure la comunitat (clau d'accés) i la ip del nostre servidor Nagios.



Si parlem d'un servidor Linux, només cal tenir instal·lat l'agent i editar un fitxer:
Per instal·lar l'agent en un servidor amb Ubuntu, només cal fer:

```
apt-get install snmpd
```

Una vegada el tenim instal·lat, editem el fitxer /etc/snmp/snmpd.conf i hauria de quedar similar al següent exemple:

```
#        Llistes        de        control        d'accés        (ACL)
##     sec.name     source     community     (alias     clau     d'accés)
com2sec local 127.0.0.1/32 la_clau_d'_accés
com2sec miredlocal 192.168.1.0/24 la_clau_d'_accés
#Se     asigna     ACL     al     grupo     de     lectura     escritura
group                MyRWGroup                v1                local
group                MyRWGroup                v2c               local
group MyRWGroup usm local
#S'assigna        ACL        al        grup        només        lectura
group               MyROGroup              v1               miredlocal
group               MyROGroup              v2c              miredlocal
group MyROGroup usm miredlocal
# Ramas MIB que se permiten ver
## name   incl/excl subtree   mask(optional)
view all  included  .1        80
# Establece permisos de lectura y escritura
## group           context   sec.model sec.level prefix read    write
notif
access MyROGroup ""       any       noauth    exact  all    none  none
access MyRWGroup ""       any       noauth    exact  all    all   all
#        Información       de       Contacto       del        Sistema
syslocation    Servidor    Linux    en    amdk6.linuxparatodos.com.mx
syscontact Administrador (fulano@algun-dominio.net)
```

Tal i com hem modificat el fitxer, hem especificat que tota el rang 192.168.1.0/24 amb la clau_d'accés pugui realitzar consultes snmp contra aquesta màquina Linux.
En cas que només volguéssim que fos una màquina la que tingui accés, només hauríem de canviar la màscara /24 per /32 i posar la ip.

2.- Realitzar un snmpwalk i exportar-ho a un fitxer

Per obtenir totes les dades possibles del dispositiu, només caldrà executar una comanda des de la consola de la nostra màquina Nagios.

Mirem doncs quina sintaxi té :

```
snmpwalk -v 2c -c clau_d'accés IP_DEL_SERVIDOR_A_MIRAR
```

Una vegada que executem aquesta comanda i donant per suposat que hem seguit tots els passos correctament, ens hauria de treure molta informació a la consolar.
El que recomano es extreure la sortida de la consola i exporta-la a un fitxer de text per poder-la llegit amb tranquil·litat:

```
snmpwalk -v 2c -c clau_d'accés IP_DEL_SERVIDOR_A_MIRAR > snmpwalkOut.txt
```

Aquesta tasca és una de les més tedioses i hem de tenir paciència per poc a poc intentar comprendre les dades que tenim.
Hem de saber esglaonar les dades que ens interessen. És recomanable en aquests cassos fer cerques dins el fitxer amb un patró concret que ens interessi.

Per exemple si volem trobar una paraula en concret, haurem d'executar una cerca dintre del nostre fitxer snmpwalkOut.txt cercant el patró en qüestió.

```
more snmpwalkOut.txt  | grep Patró_a_buscar
```

D'aquesta manera el que aconseguirem serà anar acotant la informació a analitzar i d'aquesta manera ens aproparem al valor exacte del OID que estem buscant.

3.- Analitzar i detectar els atributs que volem controlar (OID).

Per entendre l'anàlisi, el millor serà posar un exemple real com és el cas d'uns dels objectius del projecte, controlar per exemple el numero de voltes per minut que estant donant els ventiladors de l'Appliance Ironport.

Després d'haver realitzat la cerca del patró FAN he trobat la següent informació:

```
root@srv-nagga:/usr/local/nagios/libexec# snmpwalk -v 2c -c "COMUNITY"
"IP_DEL_IRONPORT" 1.3.6.1.4.1.15497.1.1.1.10
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.1 = Gauge32: 1800
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.2 = Gauge32: 4800
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.3 = Gauge32: 5025
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.4 = Gauge32: 5025
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.5 = Gauge32: 5025
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.6 = Gauge32: 4725
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.3.1 = STRING: "FAN 1"
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.3.2 = STRING: "FAN 2"
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.3.3 = STRING: "FAN 3"
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.3.4 = STRING: "FAN 4"
```

```
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.3.5 = STRING: "FAN 5"
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.3.6 = STRING: "FAN 6"
```

És fàcil veure que hi ha una correspondència entre el primer bloc de 6 objectes i el següent bloc de 6 objectes. El que vol dir el següent:

```
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.1 es el FAN 1
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.2 es el FAN 2
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.3 es el FAN 3
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.4 es el FAN 4
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.5 es el FAN 5
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.6 es el FAN 6
```

Una vegada hem arribat a aquesta conclusió sabem que el OID `SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.1` conté el numero de voltes del FAN 1.

4.- Realitzar un script en bash.

Aquesta part del desenvolupament no hauria de suposar cap molèstia per a qualsevol enginyer ja que al llarg de la carrera hem desenvolupat un munt de scripts sota bash, cshell, etc.. Per aquest motiu no donaré gaire importància al desenvolupament del mateix.

El primer que hem de fer és posicionar-nos al directori on tenim els scripts originals de l'aplicació Nagios

```
cd /usr/local/nagios/libexec
```

Des de la pròpia consola intentem fer una captura directa del valor que ens interessa:

```
snmpget -v 2c -c clan_d'accés IP_DE_L'APPLIANCE SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.1
```

El resultat ha de ser el numero de voltes que està donant el ventilador en qüestió:

```
SNMPv2-SMI::enterprises.15497.1.1.1.10.1.2.1 = Gauge32: 1800
```

Ja tenim doncs la sentencia que ens aporta el valor desitjat. Ara toca realitzar el script i el podem fer de moltes maneres diferents i totes poden ser igualment bones.

Anem a veure el script finalitzat:

```
#!/bin/bash
```

```
estado=`snmpget     -v     2c     -c     clau_d'accés     IP_IRONPORT
1.3.6.1.4.1.15497.1.1.1.10.1.2.2 | cut -c 56-60`

if [ $estado -lt 6000 ] && [ $estado -gt 100 ] ; then
        estado1="CORRECTO"
        echo $estado1 $estado rpm
        exit 0
else
        estado2="Las revoluciones del ventilador estan superando las
6000  rpm o son inferiores a 100 rpm. Posible problema "
        echo $estado1 $estado
        exit 1
fi
```

És important com sortim del script ja que determinarà l'estat a la nostra aplicació Nagios:

Exit 0 = OK
Exit 1= Warning
Exit 2= Critical
Exit 3 = Unknow

A aquest script li hem de donar permisos d'execució amb un:

```
chmod + x check_nom_de_script
```

Només ens queda comprovar que el nostre script està funcionant correctament. Per verificar-ho només cal executar-lo i veure quina resposta ens dona:

```
./check_nom_del_script
```

La resposta a l'execució d'aquesta comanda hauria de ser, segons el nostres script, haurà de ser una de les següents:

- CORRECTO XXXXX rmp
- `Las revoluciones del ventilador están superando las 6000  rpm o son inferiores a 100 rpm. Posible problema` XXXXX rmp

5.- Modificar l'aplicació Nagios per que tingui en compte els nous scripts.

Ja tenim la part complicada feta. És important doncs modificar els fitxers de configuració de Nagios per que a partir d'ara tingui en compte el nou script realitzar i el podem fer servir a qualsevol servei, host, etc...

El primer que s'ha de modificar és el fitxer: **/usr/local/nagios/etc/objects/commands.cfg**

És recomanable seguir el mateix esquema de sintaxi que la pròpia aplicació, és per això que si podem crear els scripts amb el prefixa “check_” millor.

Llavors ens quedaria així:

```
# 'check_nom_del_script' command definition
define command{
        command_name    check_nom_del_script
        command_line    $USER1$/check_nom_del_script
        }
```

Ja el podem fer servir. Ara només queda crear el servei que s’encarregarà de executar aquest script.

On posem el servei en qüestió depèn directament de la nostra estructura de fitxers.
En un principi, jo vaig classificar l’Ironport com un equip de networking.
Llavors al fitxer **/usr/local/nagios/etc/objects/network.cfg** he afegit el següent:

```
define host{
        use             generic-switch
        host_name       IRONPORT
        alias           IRONPORT
        address         IP_DEL_IRONPORT
        hostgroups      Network
        notification_interval          0
        notification_period            24x7
        notification_options           d,u,r
        }
```

I en el mateix fitxer definim el servei:

```
define service{
        use                     generic-service
        host_name               IRONPORT
        service_description     Revolucions_del_ventilador
        contact_groups          admins
        check_period                    24x7
        max_check_attempts              4
        normal_check_interval           5
        retry_check_interval            1
        notification_interval           0
        notification_period             24x7
        notification_options            c,r
        check_command           check_nom_del_script
        }
```

Sempre que modifiquem els fitxers de configuració de Nagios, per que tinguin efecte s’ha reiniciar els serveis Nagios.

```
/etc/init.d/nagios restart
```

Si hem tingut algun error al fer el script o en modificar els fitxers de Nagios ens apareixerà els errors i Nagios no podrà aixecar-se.

Per verificar que els nostres canvis han tingut èxit podem anar a la interfície web per comprovar que els nous checks s'han incorporat a la nostra aplicació:

4.3 Software de Fabricants.

A mesura que anem avançant en la complexitat dels nostres dissenys, ens trobarem amb problemes com que el propi agent del sistema operatiu no és capaç de proporcionar-nos tota la informació que ens agradaria ja que no trobem a primera vista a la sortida del snmpwalk el OID que ens interessa.

Considero apropiat comentar que si el nostre objectiu és el de controlar el hardware d'un servidor qualsevol a nivell avançat, és molt probable que l'agent snmp del propi sistema operatiu no sigui capaç de donar-nos la informació que necessitem. És per això que s'ha de tenir en compte que en aquests casos, per poder arribar als nostres objectius, haurem de fer ús d'eines addicionals.

En el nostre cas, un del objectius per poder donar solucions als nostres clients era el de poder detectar errors i degradacions dels discs que formen un RAID. Donat que disposava d'un servidor HP a l'oficina i que són molts els clients que disposen de servidors d'aquest fabricant, vaig començar a desenvolupar checks que em permetessin detectar aquestes anomalies.

Tal i com he comentat anteriorment, un dels principals problemes que vaig tenir era que la sortida del snmpwalk sencer de la màquina no portava la informació que jo necessitava per poder realitzar el script. Arribat a aquest punt, vaig començar a

investigar maneres de estendre la informació de l'agent snmp del sistema operatiu per tenir l'estat del hardware concret.

4.4 Disseny avançat de scripts.

Arribat a aquest punt, el més convenient és veure el numero màxim d'exemples reals que he anat desenvolupant. D'aquesta manera, es pot tenir una idea clara de les possibilitats que ens aporta el desenvolupament de scripting basat en snmp per tal de tenir el màxim control possible dels dispositius de la nostra xarxa. Evidentment i donat que ja hem explicat com modificar tots els paràmetres a l'aplicació, només exposaré els scripts realitzats.

IRONPORT: Appliance Relay de correu amb Antispam i antivirus.

`check_ironport_fan` → `Numero de voltes per minut del ventilador 1`
`check_ironport_MailThreads` → `Numero de Mailthreads actuals`
`check_ironport_openFilesOrSockets` → `Sockets oberts`
`check_ironport_temperature` → `temperatura a l'interior`

Nota: Aquests scripts es troben al *(Annex 8)**

Anem a veure com han quedat els nous checks dintre de l'aplicació Nagios:

IRONPORT	Files_o_sockets_oberts	OK	03-13-2009 12:23:22	5d 15h 37m 55s	1/4	CORRECTE 540 Files o Sockets Oberts
	Mail_Threads	OK	03-13-2009 12:23:23	5d 15h 37m 55s	1/4	CORRECTO 230 Mailthreads
	PING	OK	03-13-2009 12:23:16	115d 2h 2m 32s	1/4	PING OK - Packet loss = 0%, RTA = 1.81 ms
	Revolucions_del_ventilador	OK	03-13-2009 12:23:27	5d 15h 37m 51s	1/4	CORRECTO 4800 rpm
	SSH	OK	03-13-2009 12:21:12	115d 2h 0m 45s	1/4	SSH OK - OpenSSH_4.2p1 FreeBSD-20060930 (protocol 2.0)
	Temperaura_Iropnport	OK	03-13-2009 12:23:30	1d 14h 52m 44s	1/4	CORRECTE 18 graus celsuis

SERVIDOR HP: Control de hardware Avançat (Detecció de caigudes de disc dintre de RAIDS). Model Pro-Liant ML-350.

`check_raid_hp` → Control genèric de l'estat de la controladora RAID.
`check_raid_hp_disk0` → Control de l'estat del disc de la bahia 0.
`check_raid_hp_disk1` → Control de l'estat del disc de la bahia 1.

`check_raid_hp_disk2` → Control de l'estat del disc de la bahia 2.

`check_raid_hp_disk3` → Control de l'estat del disc de la bahia 3.

`check_raid_hp_disk4` → Control de l'estat del disc de la bahia 4.

`check_raid_hp_disk5` → Control de l'estat del disc de la bahia 5.

`check_raid_hp_disk` → Control de l'estat del discs a totes les bahies.

Nota: Aquests scripts es troben al *(Annex 9)**

Si volem fer un disseny òptim, la millor manera de programar un script per intentar de no saturar la màquina Nagios en cas de infraestructures molt grans amb més de 2000 checks, es poden agrupar diferents checks en un únic check.

Aquí poso un exemple d'agrupació del check_raid_hp_disk0 fins al check_raid_hp_disk5 i li dono el nom check_raid_hp_disk.

Nota: Aquests scripts es troben al *(Annex 10)**

Anem a veure com ens han quedat els nous checks:

SRV-HP	Ping	OK	03-13-2009 13:36:56	10d 20h 19m 25s	1/4	PING OK - P
	Raid_HP_GENERAL	OK	03-13-2009 13:40:44	3d 23h 50m 37s	1/4	CORRECTE
	Raid_HP_disk	OK	03-13-2009 13:41:12	10d 20h 10m 9s	1/4	CORRECTE
	Raid_HP_disk_0	OK	03-13-2009 13:37:24	10d 20h 18m 57s	1/4	CORRECTE
	Raid_HP_disk_1	OK	03-13-2009 13:38:20	10d 20h 13m 14s	1/4	CORRECTE
	Raid_HP_disk_2	OK	03-13-2009 13:37:35	10d 20h 13m 47s	1/4	CORRECTE
	Raid_HP_disk_3	OK	03-13-2009 13:37:42	8d 3h 48m 39s	1/4	CORRECTE
	Raid_HP_disk_4	OK	03-13-2009 13:37:38	10d 20h 13m 43s	1/4	CORRECTE
	Raid_HP_disk_5	OK	03-13-2009 13:36:54	10d 20h 9m 27s	1/4	CORRECTE

Aquests scripts ja complien amb èxit els meus objectius. Val a dir que si cerquem per la xarxa, podrem trobar una gran quantitat de codi que ens permet exprimir al màxim les possibilitats del protocol snmp.

Anem a veure un exemple d' un desenvolupador anomenat Jakubowski creat al 2005.

Comprovació de la càrrega de CPU que té una màquina amb Windows mitjançant el protocol snmp sense fer servir cap agent addicional.
Nota: Aquests scripts es troben al *(Annex 11)**

CAPÍTOL 5. Conclusions.

5.1 Anàlisi

El coneixement de l'estat dels dispositius que estan connectats a la xarxa on tenim el nostre sistema Nagios és el principal objectiu de qualsevol administrador de sistema que es disposa a fer una instal·lació d'aquests tipus d'aplicacions. Val a dir que arribar aquest objectiu, si es segueixen les guies que ens podem trobar per Internet, no és extremadament complicat si ens ho prenem amb calma. És per això que cal recordar que l'objectiu principal d'aquest projecte era el de fer un control avançat del hardware dels nostres dispositius fent servir el protocol snmp.

Com hem pogut veure, hem assolit els nostres objectius ja que hem trobat maneres de controlar aspectes tan concrets com el numero de voltes que dona el ventilador d'un Ironport de Cisco, l'estat dels discos d'un Raid a un servidor HP, controlar la quantitat d'espai d'un disc, etc...

Cal remarcar que tots aquests objectius s'han assolit sense haver de fer ús de cap agent addicional que ens permetés obtenir les dades que volíem. És per això que el projecte es va centrar en el protocol snmp estàndard.

Tots els desenvolupaments de scripts o aplicacions nombrats al projecte han estat comprovats per tal d'assegurar-nos que ens proporcionin els valors desitjats per tal que Nagios sigui capaç d'interpretar-los. De totes maneres, cal recordar, que tots els desenvolupaments aquí comentats estan formant part del sistema de control dels dispositius dels nostres clients que estan en producció durant 24 hores al dia i 365 dies a l'any a les nostres oficines. És per això que hem pogut comprovar que els nostres desenvolupaments no generen falsos positius i al mateix temps s'han forçat errors per comprovar que realment són efectius en cas de errors.

Arribat a aquest punt, hem de fer una crítica constructiva sobre aquest projecte analitzant els seus avantatges i els seus inconvenients.

Començarem doncs pels avantatges:

- Control de l'estat: Hem desenvolupat una eina que ens permet detectar qualsevol canvi d'estat en els dispositius que tinguem controlats, d'aquesta manera si tenim qualsevol problema ens assabentem per l'aplicació i no per la pèrdua de funcionalitat.
- Notificacions: No és necessari que estiguem tota l'estona comprovant si tenim la xara en estats correctes. La pròpia aplicació ens reporta els problemes de diferents maneres. D'aquesta manera dedicarem menys temps a l'administració de les plataformes per dedicar-lo al altres tasques més productives.
- Històric: Els estats dels dispositius i els seus canvis queden registrats i els podem consultar en qualsevol moment.
- Previsió de Catàstrofe: El fet de poder definir els llindars segons les nostres necessitats ens permet prevenir una situació crítica i evitar-la abans que succeeixi.
- Cost: No tenim costos de llicències ni i tenim suport gratuït per una comunitat d'usuaris.
- Llindars: Podem determinar llindars amb warnings per detectar situacions no crítiques però volem saber que ens estem apropant a situacions no desitjades.

Inconvenients:

- Administració Complexa: El de fer modificacions al codi de l'aplicació requereix coneixements avançats. No apte per a usuaris. El fitxers de l'aplicació tenen una relació directa el que implica que modificar un únic fitxer implica modificar varis fitxers per mantenir la coherència.
- Adaptació als nous fabricants: El fet de controlar nous dispositius passa per realitzar de nou tot el procés de detecció de OID's i trobar els significat dels valors de resposta.
- Manteniments anual: És necessari mantenir l'aplicació en constant manteniment per tal de garantir la millor cobertura dels diferents fabricants.

Anem a repassar els aspectes comercials.

Per poder garantir la rendibilitat d'aquest projecte al mercat, es van desenvolupar 2 models comercials diferents per tal d'adaptar-lo a les necessitats dels clients.

Els perfils d'autenticació que vam esmentar a l'apartat 3.4.3 s'adapten a cada als models comercials que es van dissenyar:

→ Nagios Pack. (Es fa la Instal·lació i Formació ~ 940€ /pack).
→ Manteniment Total. (Manteniment anual per 300€).

Els cost en paquet del Nagios Pack varia segons la complexitat de la infraestructura del client. És per això que desenvolupar una llista de preus de venta al públic no era aconsellable. D'aquesta manera vam optar per auditories gratuïtes de Networking prèvies als desenvolupaments per tal de determinar els dissenys.
Val a dir que un dels models comercials més estès és el de rènting o contracte mensual que equival al paquet Nagios Manteniment.

El cost total que he obtingut amb l'eina Microsoft Project és de 588 hores a un preu de 20 € ens dona un cost definitiu de **11760 €.**
Addicionalment tenim que incloure un cost de despeses de manteniment per tal d'anar adaptant i desenvolupant l'aplicació als nous possibles models de hardware dels fabricants i al mateix temps anar optimitzant l'aplicació. Aquest cost s'ha fixat en **290€**.

Anem a posar un cas exemple per tal de veure l'amortització dels esforços realitzats:
1er Any → Es venen 7 Manteniments Totals + 4 packs per Desenvolupadors.
2on Any → Es venen 6 Manteniments Totals + 3 Pack per Desenvolupadors.
3er Any → Es venen 6 Manteniments Totals + 3 Pack per Desenvolupadors.
4rt Any → Es venen 6 Manteniments Totals + 2 Pack per Desenvolupadors.
5e Any → Es venen 4 Manteniments Totals + 1 Pack per Desenvolupadors.

Any	Mantenimets acumulats	Packs acumulats	Cost Acumulat	Benefici Total
1	7	4	11.760 €	-5.860 €
2	13	7	12.050 €	-1.500 €
3	19	10	12.340 €	3.150 €
4	25	12	12.630 €	6.270 €
5	29	13	12.920 €	8.130 €

Així doncs els beneficis reals els començarem a rebre a partir del tercer any. Del quart any cap endavant els recursos destinats seran molts pocs comparats amb els beneficis.

5.2 Millores

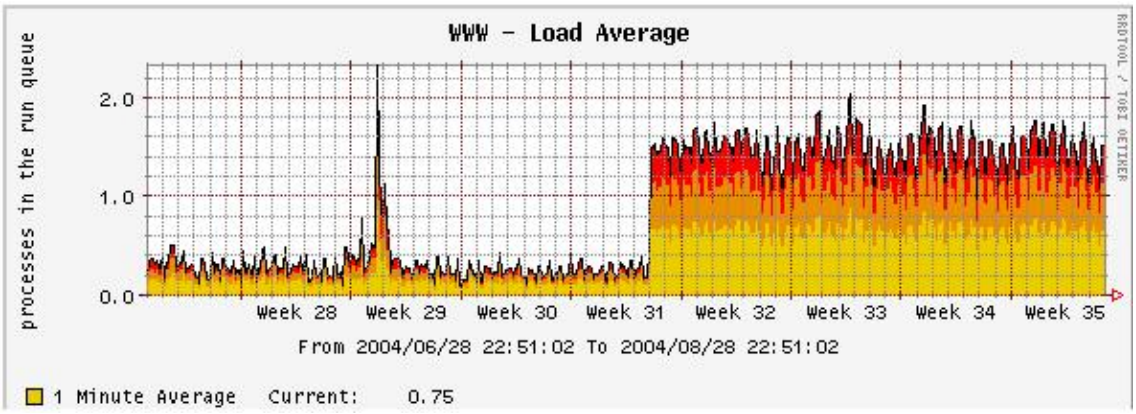
Un dels inconvenients que m'he trobat durant el desenvolupament d'aquest projecte és que m'hagués agradat al mateix temps complimentar-lo amb diferents eines per tal d'adquirir una suite d'aplicacions que ens proporcionin tot el coneixement sense gaire esforç. El problema principal que no m'ha permès capficar-me amb aquestes idees és que són aplicacions tant complexes com Nagios o més, el que significa obrir nous fronts d'investigació que m'hagués allunyat dels meus objectius.

De totes maneres, aquesta part hem permet esmentar el que considero molt interessant per tal de complimentar aquest projecte Nagios.

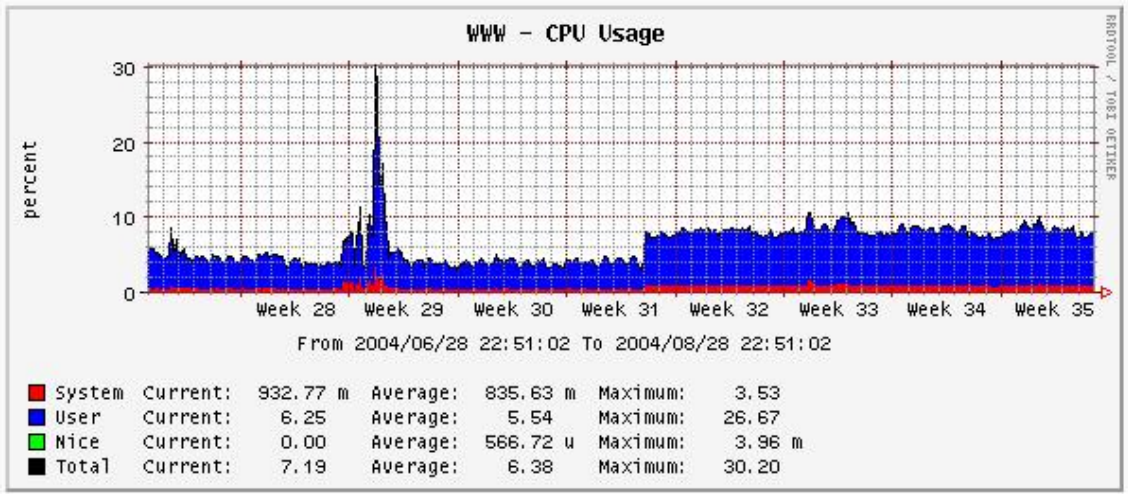
És per això que després d'haver desenvolupat aquest projecte, considero que hi ha una eina anomenada **CACTI** que ens permet fer gràfiques l'estat de la nostra xarxa, tràfic que genera permetent distingir segons el protocol, serveis, etc...

Considero que aquest eina és complementaria a Nagios ja que cada una aporta informació molt útil als administradors. Així com que Nagios només informa quan detecta que l'estat d'un valor està sota el llindar que nosaltres hem considerat crític, Cacti va guardant tots els valors a un "pooler" i va dibuixant una gràfica que mostra els valors obtinguts. D'aquesta manera és més fàcil interpretar en quins moments les nostres aplicacions pateixen estrès o simplement saber quin tipus de tràfic ens està saturant la nostra xarxa. La diferència doncs radica en que l'aplicació Nagios només sap detectar valors límits o llindars sense tenir en compte quin es el comportament normal del control que s'està realitzant i si porta molt de temps a punt de passar el llindar però encara no l'ha passat. Es per això que CACTI ens aporta més coneixements dels estats dels nostres servidors, equips, etc...

Els plugins que ens incorporen ja permeten descobrir per snmp targetes de xarxa, discs durs, cpu's etc... automàticament i seleccionem els objectes que volem dibuixar i es generen les gràfiques. Considero aquesta eina molt útil per tal de monitorar les interfícies de xarxa dels servidors o Routers.



En aquesta imatge per exemple podem veure clarament com el servidor web ha rebut moltes visites a partir de la setmana 31. Posem el cas que la infraestructura d'aquest servidor web està ben muntada i va estar sempre disponible.
Si fem una comparació entres Nagios i Cacti, amb la nostra aplicació, si no cau el servidor web no sabem la quantitat de tràfic que passa per la nostra xarxa en cada punt. El cacti ens aporta aquest valor afegit que li falta al Nagios.



Podem veure clarament que hi ha un comportament similar entre les dues imatges ja que les gràfiques son similars. Una imatge comprova la càrrega de processos del servidor i l'altra la càrrega de cpu.

Tal i com hem comentat abans, aquesta eina és el complement ideal per a la nostra plataforma.

5.3 Resum

Resum

L'objectiu d'aquest projecte és la implementació d'unes eines que ens permetin tenir el coneixement avançat del hardware o software que tenim disponible a la nostra xarxa independentment de la seva funcionalitat.

Aquest projecte intenta cobrir la necessitat de detectar errors ja siguin Hardware o Software en el funcionament dels nostres dispositius per tal de garantir accions proactives d'administració i d'aquesta manera, maximitzar la disponibilitat dels nostres serveis.

En aquesta memòria es presenten els passos a seguir per tal d'anar incorporant totes les eines necessàries al nostre sistema i així poder controlar els dispositius amb el màxim detall possible.

Resumen

El objetivo de este proyecto es la implementación de las herramientas que nos permitan tener el conocimiento avanzado del hardware o software que tenemos disponible en nuestra red independientemente de su funcionalidad.

Este proyecto intenta cubrir la necesidad de detectar errores ya sean Hardware o Software en el funcionamiento de nuestros dispositivos con tal de garantizar acciones proactivas de administración i de esta manera, maximizar la disponobilidad de nuestros servicios.

En esta memoria se presentan los pasos a seguir con tal de ir incorporando todas las herramientas necesarias a nuestra sistema i así poder controlar los dispositivos con el máximo detalle posible.

Abstract

The aim of this project is the implementation of the tools that allow us to have an advance knowledge of the hardware or software that we have available in our network regardless of its functionality.

The objective of this project is to detect whether errors as Hardware or Software can appear in the operation of our devices. Such actions will ensure a proactive management, helping us to maximize the availability of our services.

This report shows the steps to follow in order to go incorporating all the necessary tools in our system to be able to control the devices in detail.

5.4 Bibliografia

Nagios genèric:

http://www.nagios.org

http://nagiosforge.org

http://nagiosexchange.org

http://www.nagvis.org/

Srcipting:

http://es.wikipedia.org/wiki/Bash

http://es.wikipedia.org/wiki/C_Shell

Varis Add-ons per Nagios:

http://www.nagiosexchange.org/cgi-bin/page.cgi?g=Utilities%2FAddOn_Projects%2FConfiguration%2Findex.html;d=1

Script per HP

http://www.nagiosexchange.org/cgi-bin/page.cgi?g=Detailed%2F2754.html;d=1

Nagios Grapher:

http://www.nagioswiki.org/wiki/Addon:NagiosGrapher

Instal·lacions :

http://blog.evolutioncreations.com/2008/07/installing-cacti-on-ubuntu-804.html

http://oidacra.accionasolutions.net/2008/10/instalar-cacti-en-ubuntu-hardy-804/

Cacti:

http://cacti.net/

Fabricants:

http://www.hp.com

http://www.enterasys.com

http://www.cisco.com

http://www.dell.com

http://es.ts.fujitsu.com/

http://www.ibm.com

http://www.ironport.com/

http://www.netapp.com/us/

ANNEXES

0.- Synopsis default Nagios checks

Check_disk

Comprova l'espai lliure de disc d'un volum que tinguem muntat al propi servidor nagios. Té dos llindars, warning i critical.

Check_disk_smb

Aquesta comanda té un funcionament molt similar al check anterior amb la diferència que realitza la comprovació de volums compartits en equips remots.

Check_dns

Ens permet fer consultes DNS per saber la ip que correspon a un nom de domini o al inrevés (resolució inversa).

Check_ftp

Serveix per saber si el servei ftp d'un servidor està disponible i saber el seu estat.

Check_http

Aquest script ens permet comprovar els servidors http i https d'equips remots. Al mateix temps ens aporta informació com els temps de connexió, l'expiració de certificats ssl, etc...

Check_ifstatus

És un del checks que s'encarrega de controlar les interfícies de xarxa remotes. D'aquesta manera podríem guardar els valors i dibuixar gràfiques.

Check_imap

Aquest script intenta la connexió contra un servidor IMAP de correu per comprovar l'estat. Ens permet generar errors en funció de les nostres necessitats.

Check_ircd

Comprova el funcionament d'un servidor de irc remot.

Check_ldap

Aquesta comanda realitza connexions i cerques LDAP contra el servidor de la nostra xarxa que tingui aquest rol.

Check_load

Està dissenyat per controlar per tenir el coneixement de la quantitat de memòria principal que s'està fent servir al servidor nagios.

Check_log

Ens aporta la possibilitat de fer cerques de patrons als logs dels nostres sistemes.

Check_nntp

És l'encarregat d'intentar establir connexions contra un servidor remot nntp i comprovar que el servei News està actiu.

Check_nt

Aquest check ens aporta moltes funcionalitats ja que un dels paràmetres és la funcionalitat que volem saber. És un check per controlar servidors i màquines amb Windows.

Check_ntp

Executa un ntpupdate per comprovar el timestamp de la màquina local que està executant nagios.

Check_oracle

Aquesta comanada permet comprovar l'estat d'un SGBD Oracle a un pc remot així com l'estat dels trablespaces.

Check_pop

Aquesta comanda ens permet comprovar l'estat del servei POP d'un servidor remot

Check_procs

El funcionament d'aquesta comanda està pensat per ser executat al servidor on s'està executant Nagios i ens genera una avís quan es supera el llindar.

Check_real

Comprova el servei REAL d'un equip remot .

Check_smtp

És l'encarregat de veure que el servei smtp d'un servidor de correu remot està funcionant correctament. Intenta establir una connexió smtp contra el servidor.

Check_snmp

Ens permet conèixer l'estat del servei snmp d'una màquina qualsevol.

Check_ssh

És una comanada molt útil si volem saber si el servei ssh pel port 22 d'una màquina remota està escoltant correctament.

Check_swap

Ens serveix per saber la quantitat de memòria d'intercanvi que s'està fent servir i genera advertències segons els llindars.

Check_tcp

Ens permet realitzar peticions arbitràries a connexions (sockets) tcp.

Check_time

Serveix per comprovar el servei de la hora (TIME) que està funcionant a una màquina remota.

Check_udp

Al igual que check_tcp, ens permet comprovar connexions arbitràries a sockets udp

Check_ups

Ens permet controlar el servei UPS de màquines remotes.

Check_users

Ens diu el numero d'usuaris que estan connectats al nostre servidor local.

1.- Plugin Synopsis Net4Snmp

Aquí podem veure tots els checks que incorpora el plugin Net4Snmp amb una breu descripció de quines funcionalitats té cada check.

check_brocade_fan	Checks the status of a fan of a Brocade SAN Switch.
check_brocade_overall	Checks the overall status of a Brocade SAN Switch.
check_brocade_port	Checks the status of a FC interface of a Broacade SAN Switch.
check_brocade_psu	Checks the status of a power supply of a Brocade SAN Switch.
check_brocade_temp	Checks the status of a temperature sensor of a Brocade SAN Switch.
check_cisco_cpuusage	Checks the average cpu usage of a Cisco device.
check_cisco_fan	Checks the status of a fan of a Cisco device.
check_cisco_if_load	Checks the interface load of one of a Cisco device's network interfaces.
check_cisco_mem	Checks the usage of a memory pool of a Cisco device.
check_cisco_pix_conns	Checks the number of IP connections on a Cisco PIX Firewall.
check_cisco_psu	Checks the status of a power supply unit of a Cisco device.
check_cisco_temp	Checks the status of a temperature sensor of a Cisco device.
check_cisco_voltage	Checks the status of a voltage sensor of a Cisco device.
check_cisco_vpn_conns	Checks the number of connections (sessions) used on an a Cisco VPN Concentrator.
check_cisco_vpn_cpuusage	Checks the cpu usage of a Cisco VPN Concentrator.
check_cisco_vpn_fan	Checks whether a fan of Cisco VPN Concentrator has an alarm.
check_cisco_vpn_temp	Checks whether a temperature sensor of Cisco VPN Concentrator has an alarm.
check_cisco_vpn_thru	Checks the throughput utilization of a Cisco VPN Concentrator.
check_cisco_vpn_voltage	Checks whether a voltage sensor of a Cisco VPN Concentrator has an alarm.

check_cpq_fan	Checks the status of a fan of Compaq/HP server.
check_cpq_fcaaccel	Checks the status of a Compaq/HP External Array Accelerator.
check_cpq_fcaeac	Checks the status of a Compaq/HP External Array Controller.
check_cpq_fcahctrl	Checks the status of a Compaq/HPFibre Channel Host Controller.
check_cpq_fcalogdrv	Checks the status of a Compaq/HP External Array logical drive.
check_cpq_fcaphydrv	Checks the state of a physical drive in an external Compaq fibre channel array.
check_cpq_fcaspare	Checks the status of a Compaq/HP External Array spare drive.
check_cpq_ida	Checks the status of a Compaq/HP IDA Controller.
check_cpq_phydrv	Checks the state of a physical drive connected to a Compaq Intelligent Drive Array Controller.
check_cpq_temp	Checks the status of a temperature sensor of a Compaq server.
check_cpq_thermal	Checks the status of a Compaq/HP IDA Controller.
check_if_by_snmp	Checks the status of a network interface.
check_netapp_battery	Checks the NVRAM batteries of a NetApp Filer.
check_netapp_du	Checks the usage of a NetApp Filer volume.
check_netapp_fans	Checks the status of the fans of a NetApp Filer.
check_netapp_ops	Get the number of operations from a NetApp Filer.
check_netapp_overall	Checks the overall status of a NetApp Filer.
check_netapp_psus	Checks the status of the PSUs of a NetApp Filer.
check_netapp_raiddrv	Checks the state of a physical drive of a NetApp Filer.
check_netapp_spare	Checks the state of a spare drive of a NetApp Filer.
check_netapp_temp	Checks the temperature status of a NetApp Filer.
check_netapp_vol	Checks the status of a volume of a NetApp Filer.
check_storage_by_snmp	Checks the usage of some storage device (disk, ram, …).
check_ucd_lms_fan	Checks the value of a fan speed sensor which is monitored using lm_sensors.
check_ucd_lms_temp	Checks a temperature which is monitored using lm_sensors.
check_ucd_lms_voltage	Checks the voltage measured with a sensor which is monitored using lm_sensors.
check_ucd_snmp_cpu	Reports the cpu usage of a computer using the UCD-SNMP-MIB.
check_ucd_snmp_load	Checks the average system loads of a computer using the UCD-SNMP-MIB.
check_ucd_snmp_mem	Checks the memory usages of a computer using the UCD-SNMP-MIB.
check_ups_alarms	Checks if an UPS has any present alarms.
check_ups_battery	Checks the battery status of an UPS.

check_ups_bypass	Checks the status of an UPS bypass line.
check_ups_input	Checks the status of an UPS input line.
check_ups_output	Checks the status of an UPS output line.
check_ups_outputs	Checks the source of the output of an UPS.
check_winf_cpuusage	Reports the cpu usage of a Microsoft Windows computer using SNMP Informant.
check_winf_mem	Checks the memory usage of a Microsoft Windows computer using SNMP Informant.

2.- Extracte del fitxer de configuració: cgi.cfg

Fitxer de configuració de l'aplicació Nagios que està ubicat al directori per defecte: /usr/local/nagios/etc/cgi.cfg a partir del versions 3.X.

```
##############################################################################
#
# CGI.CFG - Sample CGI Configuration File for Nagios 3.0.1
#
# Last Modified: 10-07-2007
#
##############################################################################


# MAIN CONFIGURATION FILE
# This tells the CGIs where to find your main configuration file.
# The CGIs will read the main and host config files for any other
# data they might need.

main_config_file=/usr/local/nagios/etc/nagios.cfg



# PHYSICAL HTML PATH
# This is the path where the HTML files for Nagios reside.  This
# value is used to locate the logo images needed by the statusmap
# and statuswrl CGIs.

physical_html_path=/usr/local/nagios/share



# URL HTML PATH
# This is the path portion of the URL that corresponds to the
# physical location of the Nagios HTML files (as defined above).
# This value is used by the CGIs to locate the online documentation
# and graphics.  If you access the Nagios pages with an URL like
# http://www.myhost.com/nagios, this value should be '/nagios'
# (without the quotes).

url_html_path=/nagios
```

```
# CONTEXT-SENSITIVE HELP
# This option determines whether or not a context-sensitive
# help icon will be displayed for most of the CGIs.
# Values: 0 = disables context-sensitive help
#         1 = enables context-sensitive help

show_context_help=0



# PENDING STATES OPTION
# This option determines what states should be displayed in the web
# interface for hosts/services that have not yet been checked.
# Values: 0 = leave hosts/services that have not been check yet in
their original state
#         1 = mark hosts/services that have not been checked yet as
PENDING

use_pending_states=1




# AUTHENTICATION USAGE
# This option controls whether or not the CGIs will use any
# authentication when displaying host and service information, as
# well as committing commands to Nagios for processing.
#
# Read the HTML documentation to learn how the authorization works!
#
# NOTE: It is a really *bad* idea to disable authorization, unless
# you plan on removing the command CGI (cmd.cgi)!  Failure to do
# so will leave you wide open to kiddies messing with Nagios and
# possibly hitting you with a denial of service attack by filling up
# your drive by continuously writing to your command file!
#
# Setting this value to 0 will cause the CGIs to *not* use
# authentication (bad idea), while any other value will make them
# use the authentication functions (the default).

use_authentication=1



# DEFAULT USER
# Setting this variable will define a default user name that can
# access pages without authentication.  This allows people within a
# secure domain (i.e., behind a firewall) to see the current status
# without authenticating.  You may want to use this to avoid basic
# authentication if you are not using a secure server since basic
# authentication transmits passwords in the clear.
#
# Important:  Do not define a default username unless you are
# running a secure web server and are sure that everyone who has
# access to the CGIs has been authenticated in some manner!  If you
# define this variable, anyone who has not authenticated to the web
# server will inherit all rights you assign to this user!

#default_user_name=guest
```

3.- Extracte del fitxer de configuració nagios.cfg

Fitxer de configuració de l'aplicació Nagios que està ubicat al directori per defecte: /usr/local/nagios/etc/nagios.cfg a partir del versions 3.X.

```
##############################################################################
#
# NAGIOS.CFG - Sample Main Config File for Nagios 3.0.1
#
# Read the documentation for more information on this configuration
# file.  I've provided some comments here, but things may not be so
# clear without further explanation.
#
# Last Modified: 12-14-2007
#
##############################################################################


# LOG FILE
# This is the main log file where service and host events are logged
# for historical purposes.  This should be the first option specified
# in the config file!!!
log_file=/usr/local/nagios/var/nagios.log

# OBJECT CONFIGURATION FILE(S)
# These are the object configuration files in which you define hosts,
# host groups, contacts, contact groups, services, etc.
# You can split your object definitions across several config files
# if you wish (as shown below), or keep them all in a single config file.

# You can specify individual object config files as shown below:
cfg_file=/usr/local/nagios/etc/objects/commands.cfg
cfg_file=/usr/local/nagios/etc/objects/contacts.cfg
cfg_file=/usr/local/nagios/etc/objects/timeperiods.cfg
cfg_file=/usr/local/nagios/etc/objects/templates.cfg
# Definitions for monitoring the local (Linux) host
cfg_file=/usr/local/nagios/etc/objects/linux.cfg
```

```
   # Definitions for monitoring a Windows machine
   cfg_file=/usr/local/nagios/etc/objects/windowsComtech.cfg
   # Definitions for monitoring a VMWARE machine
   cfg_file=/usr/local/nagios/etc/objects/vmwareservers.cfg
   # Definitions for monitoring a router/switch/Network element
   cfg_file=/usr/local/nagios/etc/objects/network.cfg
```

4.- Fitxer de configuració Contacts.cfg

```
##############################################################################
##############################################################################
#
# CONTACTS
#
##############################################################################
##############################################################################
# Just one contact defined by default - the Nagios admin (that's you)
# This contact definition inherits a lot of default values from the 'generic-
contact'
# template which is defined elsewhere.

define contact{
        contact_name                    nagiosadmin             ;
        use                             generic-contact         ;
        alias                           Nagios Admin            ;
        service_notification_options    w,u,c,r                 ;
        host_notification_options       d,u,r                   ;
        email                           x1@comtech.es           ;
        }

define contact{
        contact_name                    nagiosadmin2            ;
        use                             generic-contact         ;
        alias                           Nagios Admin            ;
        service_notification_options    w,u,c,r                 ;
        host_notification_options       d,u,r                   ;
```

```
        email                           x2@comtech.es ;
        }

define contact{
        contact_name                    nagiosadmin3            ;
        use                             generic-contact         ;
        alias                           Nagios Admin            ;
        service_notification_options    w,u,c,r                 ;
        host_notification_options       d,u,r                   ;
        email                           x3@comtech.es ;
        }

define contact{
        contact_name                    nagiosadmin4            ;
        use                             generic-contact         ;
        alias                           Nagios Admin            ;
        service_notification_options    w,u,c,r                 ;
        host_notification_options       d,u,r                   ;
        email                           x4@comtech.es ;
        }


################################################################################
################################################################################
#
# CONTACT GROUPS
#
################################################################################
################################################################################

# We only have one contact in this simple configuration file, so there is
# no need to create more than one contact group.


define contactgroup{
        contactgroup_name  admins
        alias              Nagios Administrators
        members            nagiosadmin, nagiosadmin2, nagiosadmin3,nagiosadmin4
        }
```

5.- Configuració Notificació

```
define host{
        use             windows-server  ;
        host_name       SRV-SQL-01      ;
        alias           SRV-SQL-01      ;
        address         192.168.100.XX  ;
        notification_interval          0
        notification_period            24x7
        notification_options           d,u,r
        }
```

```
################################################################################
#
# SERVICES SRV-SQL-01
#
################################################################################

# Create a service for monitoring the version of NSCLient++ that is installed
# Change the host_name to match the name of the host you defined above

define service{
        use                     generic-service
        host_name               SRV-SQL-01
        service_description     Memory Usage
        contact_groups          admins
        check_period                    24x7
        max_check_attempts              4
        normal_check_interval           5
        retry_check_interval            1
        notification_interval           0
        notification_period             24x7
        notification_options            c,r
        check_command           check_nt!MEMUSE!-w 80 -c 90
        }
```

```
# Create a service for monitoring C:\ disk usage
# Change the host_name to match the name of the host you defined above

define service{
        use                     generic-service
        host_name               SRV-SQL-01
        service_description     C:\ Drive Space
        contact_groups          admins
        check_period                    24x7
        max_check_attempts              4
        normal_check_interval           5
        retry_check_interval            1
        notification_interval           0
        notification_period             24x7
        notification_options            c,r
        check_command           check_nt!USEDDISKSPACE!-l c -w 80 -c 90
```

6.- Configuració Postfix

```
# See /usr/share/postfix/main.cf.dist for a commented, more complete
version

# Debian specific:  Specifying a file name will cause the first
# line of that file to be used as the name.  The Debian default
# is /etc/mailname.
#myorigin = /etc/mailname

smtpd_banner = $myhostname ESMTP $mail_name (Debian/GNU)
biff = no

# appending .domain is the MUA's job.
append_dot_mydomain = no

# Uncomment the next line to generate "delayed mail" warnings
#delay_warning_time = 4h

# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_use_tls=yes
smtpd_tls_session_cache_database =
```

```
btree:${queue_directory}/smtpd_scache
smtp_tls_session_cache_database = btree:${queue_directory}/smtp_scache

# See /usr/share/doc/postfix/TLS_README.gz in the postfix-doc package
for
# information on enabling SSL in the smtp client.

myhostname = srv-nagios.comtech.local
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
myorigin = /etc/mailname
mydestination = srv-nagios.comtech.local, localhost.comtech.local, ,
localhost
relayhost = IP_DEL_SERVIDOR_DE_CORREU_CORPORATIU
mynetworks = 127.0.0.0/8
mailbox_command = procmail -a "$EXTENSION"
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
```

7.- Fixer de configuració cgi.cfg

```
##############################################################################
#
# CGI.CFG - Sample CGI Configuration File for Nagios 3.0.1
#
# Last Modified: 10-07-2007
#
##############################################################################


# MAIN CONFIGURATION FILE
# This tells the CGIs where to find your main configuration file.
# The CGIs will read the main and host config files for any other
# data they might need.

main_config_file=/usr/local/nagios/etc/nagios.cfg


# PHYSICAL HTML PATH
# This is the path where the HTML files for Nagios reside.  This
# value is used to locate the logo images needed by the statusmap
# and statuswrl CGIs.

physical_html_path=/usr/local/nagios/share

# URL HTML PATH
# This is the path portion of the URL that corresponds to the
```

```
# physical location of the Nagios HTML files (as defined above).
# This value is used by the CGIs to locate the online documentation
# and graphics.  If you access the Nagios pages with an URL like
# http://www.myhost.com/nagios, this value should be '/nagios'
# (without the quotes).

url_html_path=/nagios

# CONTEXT-SENSITIVE HELP
# This option determines whether or not a context-sensitive
# help icon will be displayed for most of the CGIs.
# Values: 0 = disables context-sensitive help
#         1 = enables context-sensitive help

show_context_help=0

# PENDING STATES OPTION
# This option determines what states should be displayed in the web
# interface for hosts/services that have not yet been checked.
# Values: 0 = leave hosts/services that have not been check yet in
their original state
#         1 = mark hosts/services that have not been checked yet as
PENDING

use_pending_states=1


# AUTHENTICATION USAGE
# This option controls whether or not the CGIs will use any
# authentication when displaying host and service information, as
# well as committing commands to Nagios for processing.
#
# Read the HTML documentation to learn how the authorization works!
#
# NOTE: It is a really *bad* idea to disable authorization, unless
# you plan on removing the command CGI (cmd.cgi)!  Failure to do
# so will leave you wide open to kiddies messing with Nagios and
# possibly hitting you with a denial of service attack by filling up
# your drive by continuously writing to your command file!
#
# Setting this value to 0 will cause the CGIs to *not* use
# authentication (bad idea), while any other value will make them
# use the authentication functions (the default).

use_authentication=1

# DEFAULT USER
# Setting this variable will define a default user name that can
# access pages without authentication.  This allows people within a
# secure domain (i.e., behind a firewall) to see the current status
# without authenticating.  You may want to use this to avoid basic
# authentication if you are not using a secure server since basic
# authentication transmits passwords in the clear.
#
# Important:  Do not define a default username unless you are
# running a secure web server and are sure that everyone who has
# access to the CGIs has been authenticated in some manner!  If you
# define this variable, anyone who has not authenticated to the web
# server will inherit all rights you assign to this user!

#default_user_name=guest
```

```
# SYSTEM/PROCESS INFORMATION ACCESS
# This option is a comma-delimited list of all usernames that
# have access to viewing the Nagios process information as
# provided by the Extended Information CGI (extinfo.cgi).  By
# default, *no one* has access to this unless you choose to
# not use authorization.  You may use an asterisk (*) to
# authorize any user who has authenticated to the web server.

authorized_for_system_information=nagiosadmin

# CONFIGURATION INFORMATION ACCESS
# This option is a comma-delimited list of all usernames that
# can view ALL configuration information (hosts, commands, etc).
# By default, users can only view configuration information
# for the hosts and services they are contacts for. You may use
# an asterisk (*) to authorize any user who has authenticated
# to the web server.

authorized_for_configuration_information=nagiosadmin

# SYSTEM/PROCESS COMMAND ACCESS
# This option is a comma-delimited list of all usernames that
# can issue shutdown and restart commands to Nagios via the
# command CGI (cmd.cgi).  Users in this list can also change
# the program mode to active or standby. By default, *no one*
# has access to this unless you choose to not use authorization.
# You may use an asterisk (*) to authorize any user who has
# authenticated to the web server.

authorized_for_system_commands=nagiosadmin

# GLOBAL HOST/SERVICE VIEW ACCESS
# These two options are comma-delimited lists of all usernames that
# can view information for all hosts and services that are being
# monitored.  By default, users can only view information
# for hosts or services that they are contacts for (unless you
# you choose to not use authorization). You may use an asterisk (*)
# to authorize any user who has authenticated to the web server.

authorized_for_all_services=nagiosadmin
authorized_for_all_hosts=nagiosadmin

# GLOBAL HOST/SERVICE COMMAND ACCESS
# These two options are comma-delimited lists of all usernames that
# can issue host or service related commands via the command
# CGI (cmd.cgi) for all hosts and services that are being monitored.
# By default, users can only issue commands for hosts or services
# that they are contacts for (unless you you choose to not use
# authorization).  You may use an asterisk (*) to authorize any
# user who has authenticated to the web server.

authorized_for_all_service_commands=nagiosadmin
authorized_for_all_host_commands=nagiosadmin

# STATUSMAP BACKGROUND IMAGE
# This option allows you to specify an image to be used as a
# background in the statusmap CGI.  It is assumed that the image
# resides in the HTML images path (i.e.
```

```
/usr/local/nagios/share/images).
# This path is automatically determined by appending "/images"
# to the path specified by the 'physical_html_path' directive.
# Note:  The image file may be in GIF, PNG, JPEG, or GD2 format.
# However, I recommend that you convert your image to GD2 format
# (uncompressed), as this will cause less CPU load when the CGI
# generates the image.

#statusmap_background_image=smbackground.gd2

# DEFAULT STATUSMAP LAYOUT METHOD
# This option allows you to specify the default layout method
# the statusmap CGI should use for drawing hosts.  If you do
# not use this option, the default is to use user-defined
# coordinates.  Valid options are as follows:
#       0 = User-defined coordinates
#       1 = Depth layers
#       2 = Collapsed tree
#       3 = Balanced tree
#       4 = Circular
#       5 = Circular (Marked Up)

default_statusmap_layout=5

# DEFAULT STATUSWRL LAYOUT METHOD
# This option allows you to specify the default layout method
# the statuswrl (VRML) CGI should use for drawing hosts.  If you
# do not use this option, the default is to use user-defined
# coordinates.  Valid options are as follows:
#       0 = User-defined coordinates
#       2 = Collapsed tree
#       3 = Balanced tree
#       4 = Circular

default_statuswrl_layout=4

# STATUSWRL INCLUDE
# This option allows you to include your own objects in the
# generated VRML world.  It is assumed that the file
# resides in the HTML path (i.e. /usr/local/nagios/share).

#statuswrl_include=myworld.wrl

# PING SYNTAX
# This option determines what syntax should be used when
# attempting to ping a host from the WAP interface (using
# the statuswml CGI.  You must include the full path to
# the ping binary, along with all required options.  The
# $HOSTADDRESS$ macro is substituted with the address of
# the host before the command is executed.
# Please note that the syntax for the ping binary is
# notorious for being different on virtually ever *NIX
# OS and distribution, so you may have to tweak this to
# work on your system.

ping_syntax=/bin/ping -n -U -c 5 $HOSTADDRESS$

# REFRESH RATE
# This option allows you to specify the refresh rate in seconds
# of various CGIs (status, statusmap, extinfo, and outages).
```

```
refresh_rate=90



# ESCAPE HTML TAGS
# This option determines whether HTML tags in host and service
# status output is escaped in the web interface.  If enabled,
# your plugin output will not be able to contain clickable links.

escape_html_tags=1

# SOUND OPTIONS
# These options allow you to specify an optional audio file
# that should be played in your browser window when there are
# problems on the network.  The audio files are used only in
# the status CGI.  Only the sound for the most critical problem
# will be played.  Order of importance (higher to lower) is as
# follows: unreachable hosts, down hosts, critical services,
# warning services, and unknown services. If there are no
# visible problems, the sound file optionally specified by
# 'normal_sound' variable will be played.
#
#
# <varname>=<sound_file>
#
# Note: All audio files must be placed in the /media subdirectory
# under the HTML path (i.e. /usr/local/nagios/share/media/).

#host_unreachable_sound=hostdown.wav
#host_down_sound=hostdown.wav
#service_critical_sound=critical.wav
#service_warning_sound=warning.wav
#service_unknown_sound=warning.wav
#normal_sound=noproblem.wav

# URL TARGET FRAMES
# These options determine the target frames in which notes and
# action URLs will open.

action_url_target=_blank
notes_url_target=_blank

# LOCK AUTHOR NAMES OPTION
# This option determines whether users can change the author name
# when submitting comments, scheduling downtime.  If disabled, the
# author names will be locked into their contact name, as defined in
Nagios.
# Values: 0 = allow editing author names
#         1 = lock author names (disallow editing)

lock_author_names=1

# SPLUNK INTEGRATION OPTIONS
# These options allow you to enable integration with Splunk
# in the web interface.  If enabled, you'll be presented with
# "Splunk It" links in various places in the CGIs (log file,
# alert history, host/service detail, etc).  Useful if you're
# trying to research why a particular problem occurred.
# For more information on Splunk, visit http://www.splunk.com/

# This option determines whether the Splunk integration is enabled
```

```
# Values: 0 = disable Splunk integration
#         1 = enable Splunk integration

#enable_splunk_integration=1


# This option should be the URL used to access your instance of Splunk
```

8.- Scipts de comprovació Appliance IronPort de Cisco.

check_ironport_fan

```
estado=`snmpget -v 2c -c clau 192.168.X.Y
1.3.6.1.4.1.15497.1.1.1.10.1.2.2 | cut -c 56-60`

if [ $estado -lt 6000 ] && [ $estado -gt 100 ] ; then
        estado1="CORRECTO"
        echo $estado1 $estado rpm
        exit 0
else
        estado2="Las revoluciones del ventilador estan superando las
6000  rpm o son inferiores a 100 rpm. Posible problema "
        echo $estado1 $estado
        exit 1
fi
```

check_ironport_MailThreads

```
#!/bin/bash

estado=`snmpget -v 2c -c clau 192.168.X.Y 1.3.6.1.4.1.15497.1.1.1.20.0
| cut -c 52-60`

if [ $estado -lt 1000 ] && [ $estado -gt 1 ] ; then
        estado1="CORRECTO"
        echo $estado1 $estado Mailthreads
        exit 0
else
        estado2="El nombre de threads es superior a 1000 o son
inferiors a 1. Possible problema "
        echo $estado1 $estado
        exit 1
```

check_ironport_openFilesOrSockets

```
#!/bin/bash
```

```
estado=`snmpget -v 2c -c clau 192.168.X.Y 1.3.6.1.4.1.15497.1.1.1.19.0
| cut -c 52-60`

if [ $estado -lt 1000 ] && [ $estado -gt 0 ] ; then
        estado1="CORRECTE"
        echo $estado1 $estado Files o Sockets Oberts
        exit 0
else
        estado2="El nombre de Files o Sockets oberts es superior a
1000 o 0 . Posible problema "
        echo $estado1 $estado
        exit 1
fi
```

check_ironport_temperature

```
#!/bin/bash

estado=`snmpget -v 2c -c comtech 192.168.100.231
1.3.6.1.4.1.15497.1.1.1.9.1.2.1 | cut -c 56-60`

if [ $estado -lt 26 ] && [ $estado -gt 14 ] ; then
        estado1="CORRECTE"
        echo $estado1 $estado graus celsuis
        exit 0
else
        estado2="La temperatura es superior a 26 graus o inferior a
14. Possible problema "
        echo $estado1 $estado
        exit 1
fi
```

9.- Scipts de comprovació RAID's del servidor HP ML 350.

check_raid_hp

```
#!/bin/bash

server=$1
estado=`snmpwalk -v 2c -c comtech $server 1.3.6.1.4.1.232.3.1.3 | cut
-c 48`

if [ $estado = 2  ] ; then
        estado1="CORRECTE"
        echo $estado1
        exit 0
else
        estado2="El estado de los RAIDS es DEGRADED o FAILED"
        echo $estado2
        exit 2
fi
```

check_raid_hp_disk0

```
#!/bin/bash

server=$1
estado=`snmpwalk -v 2c -c comtech $server
.1.3.6.1.4.1.232.3.2.5.1.1.6.3.128 | cut -c 58`

if [ $estado = 2  ] ; then
        estado1="CORRECTE"
        echo $estado1
        exit 0
else
        estado2="El estado del disco de la bahia 0 es DEGRADED o
FAILED"
        echo $estado2
        exit 2
fi
```

check_raid_hp_disk1

```
#!/bin/bash

server=$1
estado=`snmpwalk -v 2c -c comtech $server
.1.3.6.1.4.1.232.3.2.5.1.1.6.3.129 | cut -c 58`

if [ $estado = 2  ] ; then
        estado1="CORRECTE"
        echo $estado1
        exit 0
else
        estado2="El estado del disco de la bahia 1 es DEGRADED o
FAILED"
        echo $estado2
        exit 2
fi
```

check_raid_hp_disk2

```
#!/bin/bash

server=$1
estado=`snmpwalk -v 2c -c comtech $server
.1.3.6.1.4.1.232.3.2.5.1.1.6.3.130 | cut -c 58`

if [ $estado = 2  ] ; then
        estado1="CORRECTE"
        echo $estado1
        exit 0
else
        estado2="El estado del disco de la bahia 2 es DEGRADED o
```

```
FAILED"
        echo $estado2
        exit 2
fi
```

check_raid_hp_disk3

```
#!/bin/bash

server=$1
estado=`snmpwalk -v 2c -c comtech $server
.1.3.6.1.4.1.232.3.2.5.1.1.6.3.131 | cut -c 58`

if [ $estado = 2  ] ; then
        estado1="CORRECTE"
        echo $estado1
        exit 0
else
        estado2="El estado del disco de la bahia 3 es DEGRADED o
FAILED"
        echo $estado2
        exit 2
fi
```

check_raid_hp_disk4

```
#!/bin/bash

server=$1
estado=`snmpwalk -v 2c -c comtech $server
.1.3.6.1.4.1.232.3.2.5.1.1.6.3.132 | cut -c 58`

if [ $estado = 2  ] ; then
        estado1="CORRECTE"
        echo $estado1
        exit 0
else
        estado2="El estado del disco de la bahia 4 es DEGRADED o
FAILED"
        echo $estado2
        exit 2
fi
```

check_raid_hp_disk5

```
#!/bin/bash

server=$1
estado=`snmpwalk -v 2c -c comtech $server
.1.3.6.1.4.1.232.3.2.5.1.1.6.3.133 | cut -c 58`

if [ $estado = 2  ] ; then
```

```
        estado1="CORRECTE"
        echo $estado1
        exit 0
else
        estado2="El estado del disco de la bahia 5 es DEGRADED o
FAILED"
        echo $estado2
        exit 2
fi
```

10.- Scipt de optimitzat de comprovació RAID's del servidor HP ML 350.

check_raid_hp_disk

```
#!/bin/bash

server=$1
oid=".1.3.6.1.4.1.232.3.2.5.1.1.6.3."
sDisk="El disco de la bahia "
i=0

for iDisk in 128 129 130 131 132 133
do
   estado=`snmpwalk -v 2c -c comtech $server
.1.3.6.1.4.1.232.3.2.5.1.1.6.3.${iDisk} | awk -F'INTEGER:' '{print
$2}'`
   if [ $estado = 2  ] ; then
        estado1="CORRECTE"
   else
      sDisk="$sDisk $i, "
   fi
let i=$i+1
done
if [ "${sDisk}" == "El disco de la bahia " ]
then
  echo "CORRECTE"
  exit 0
else
  echo "$sDisk esta con error"
  exit 2
fi
```

11.- Scipt de comprovació CPU d'un servidor windows.

check_win_snmp_cpuload.pl

```
#!/usr/bin/perl
```

```
# Author : jakubowski Benjamin
# Date : 19/12/2005
# check_win_snmp_cpuload.pl IP COMMUNITY PORT warning critical
# Update : 08/06/2007
# Author : Eric. P
# Print  : CPULOAD


sub print_usage {
print "check_win_snmp_cpuload.pl IP COMMUNITY warning critical\n";
}

$PROGNAME = "check_win_snmp_cpuload.pl";

if ( @ARGV[0] eq "" || @ARGV[1] eq "" || @ARGV[2] eq "" ) {
print_usage();
exit 0;
}

$STATE_CRITICAL = 2;
$STATE_WARNING = 1;
$STATE_UNKNONW = 3;

$STATE_OK = 0;

$IP=@ARGV[0];
$COMMUNITY=@ARGV[1];
$warning=@ARGV[2];
$critical=@ARGV[3];
$resultat =`snmpwalk -v 1 -c $COMMUNITY $IP 1.3.6.1.2.1.25.3.3.1.2`;
if ( $resultat ) {
@pourcentage = split (/\n/,$resultat);
$i=0;
foreach ( @pourcentage ) {
s/HOST-RESOURCES-MIB::hrProcessorLoad.\d+ = INTEGER://g;
$use_total+=$_;
$i++;
}
$use = $use_total / $i ;

if ( $use < $warning ) {
print "OK : CPU load $use%\n";

##### AJOUT ICI ######
print "|cpu_load=".$use."%".";".$warning.";".$critic al;
####################

exit $STATE_OK;
} elsif ( $use < $critical ) {
print "WARNING : CPU load $use%\n";
exit $STATE_WARNING;
} else {
print "CRITICAL : CPU load :$use%\n";
exit $STATE_CRITICAL;
}
} else {
print "Unkonwn : No response\n";
exit $STATE_UNKNONW;
}
```