UNIVERSITE POLYTECHNIQUE
DE BOBO DIOULASSO
(U.P.B)

.....

ECOLE SUPERIEURE D'INFORMATIQUE
(E.S.I)

E.S.I

MINISTERE DE LA FONCTION
PUBLIQUUE ET DE LA REFORME
DE L'ETAT

.....

CAISSE AUTONOME DE RETRAITE
DES FONCTIONNAIRES
(CARFO)

Mémoire de fin de cycle

THEME :

ETUDE ET PROPOSITION D'UNE SOLUTION DE SUPERVISION RESEAU BASE SUR LE LOGICIEL LIBRE NAGIOS

Présenté par : HIEN K. Rodrigue Romaric

Pour l'obtention du diplôme d'ingénieur des Travaux Informatiques
Option Réseaux et Maintenance Informatique

Période allant du 2 Août 2010 au 2 Novembre 2010

Maître de stage:
Monsieur BADO Noël
Ingénieur informaticien
Chef de la Cellule informatique
de la CARFO

Superviseur:
Dr PODA Pasteur
Enseignant à l'E.S.I

Année universitaire 2009 – 2010

DEDICACE

Ce document est dédié à ma MAMAN CHERIE, Madame HIEN/SOME Imambo, et à Madame SOME/HIEN Éveline qui m'ont toujours soutenu et encouragé.

En la mémoire de mon PERE, HIEN Nigoura Galip, que DIEU a rappelé à lui le 09 janvier 2000. Que la terre lui soit légère et que son âme repose en paix.

Remerciements

Mes sincères remerciements s'adressent à :

- DIEU TOUT PUISSANT, qui, par son œuvre divine m'a donné la santé, la force et le courage pour réussir dans mes études.
- L'ECOLE SUPERIEURE D'INFORMATIQUE, pour la qualité de la formation, à travers elle tout le corps enseignant et administratif.
- Monsieur le DIRECTEUR GENERAL de la CARFO qui a bien voulu m'accepter au sein de sa structure.
- Monsieur Noël BADO, Ingénieur informaticien, DIRECTEUR de la CELLULE INFORMATIQUE de la CARFO, mon maître de stage, pour sa disponibilité, son aide et ses conseils.
- Dr Pasteur PODA, enseignant à l'ESI, mon superviseur, pour sa disponibilité et ses remarques pertinentes.
- Monsieur François D'Assise OUATTARA, grâce à lui le stage a été possible.
- Monsieur Jean Daniel DABIRE, mon parrain, pour ses sages conseils.
- Monsieur HIEN Nazaire, ingénieur informaticien, pour son soutien.
- La famille SOME, pour son soutien moral, matériel et financier.
- Mes Frères et Sœur, pour leur soutien moral.
- Mes Camarades de classe, pour l'amitié et la bonne collaboration.
- Mes amis, dont certains ont été des Frères pour moi.

Avant propos

L'Ecole Supérieure d'Informatique (ESI) est un établissement d'enseignement supérieur et de recherche de l'Université Polytechnique de BOBO-DIOULASSO (UPB).
Créée en 1991 dans le but de ne pas être en marge de l'évolution des technologies de l'information et de la communication (TIC), l'ESI est la seule école supérieure d'informatique publique du pays à former des ingénieurs de conception informatique, des ingénieurs de travaux informatiques et des étudiants de niveau DEA informatique. L'ESI est l'un des six établissements que compte l'Université Polytechnique de BOBO-DIOULASSO.
Afin de les préparer à une meilleure insertion dans le monde professionnel, l'Ecole Supérieure d'Informatique, contraint ses étudiants à effectuer un stage pratique obligatoire de douze (12) semaines en plus des acquis théoriques, qui est sanctionné par une soutenance publique à l'enceinte de l'université.
C'est dans cette optique que nous avons effectué un stage, du 2 Août au 2 Novembre 2010, au sein de la cellule informatique de la CARFO avec pour thème : **« Etude et proposition d'une solution de supervision du réseau informatique basé sur le logiciel libre NAGIOS ».**

Sommaire

DEDICACE 1

Remerciements 2

Avant propos 3

Sommaire 4

Sigles et abréviations 6

Introduction Générale 7

Chapitre 1 : Présentation de la CARFO et de son réseau informatique. 8

Introduction 8

1.1Historique 8

1.2Missions et attributions 9

1.3Orientations stratégiques 10

1.4 Organigramme de la CARFO 10

1.5 Le réseau informatique 11

Conclusion 14

Chapitre 2 : La supervision informatique 15

Introduction 15

2.1 Objectifs et avantages 15

2.2 Méthodes de supervision informatique 15

2.3 Le protocole SNMP 16

2.4 Les moniteurs de supervisions 20

Conclusion 22

Chapitre 3 : Etude de Nagios 23

Introduction 23

3.1 Historique 23

3.2 Fonctionnalités 23

3.3 Architecture 24

3.4 Les principes de base de Nagios 29
3.6 Installation 35
3.7 Configuration 36
3.8 La supervision des éléments du réseau 38
3.9 Les types de supervision 40
Conclusion 43
Chapitre 4 : proposition de la solution de supervision et mise en œuvre 44
Introduction 44
4.1 Inventaire des équipements du réseau 44
4.2 Détermination des objets à prendre en compte pour la supervision 45
4.3 Regroupement des objets en entités logique 46
4.4 Proposition de la solution de supervision 48
4.5 Les besoins 53
4.6 Mise en œuvre 53
Conclusion 67
Conclusion générale 69
Bibliographie et webographie 70
Table des matières 71
Annexe 75

Sigles et abréviations

CGI: Common Gateway Interface

CMIP: Common Management Information Protocol

DES: Data Encryption Standard

GPL: General Public License

HTTP: HyperText Transfer Protocol

ICMP: Internet Control Message Protocol

IETF: Internet Engineering Task Force

IP: Internet Protocol

ISO: International Standard Organization

LAN: Local Area Network

MAC: Media Access Control

MAN: Metropolitans Area Network

MIB: Management Information Base

NAC: Network Admission Control

NMA: Network Manager Application

NMS: Network Management Station

NRPE: Nagios Remote Plugin Executor

NSCA: Nagios Service Check Acceptor

OID: Object Identifier

PC: Personal Computer

PDU: Packet Data Unit

POP: Post Office Protocol

SGMP: Simple Gateway Manager Protocol

SMI: Structure of Management Information

SMS: Short Message Service

SNMP: Simple Network Management Protocol

SSL: Socket Secure Layer

TCP: Transmission Control Protocol

UDP: User Datagram Protocol

WAN: Wide Area Network

Introduction Générale

Seul établissement dans le pays des Hommes Intègres, à s'occuper des retraités de la fonction publique, la CARFO (Caisse autonome de retraite des fonctionnaires), est depuis longtemps bien intégrée dans le quotidien de tous les travailleurs de l'administration publique. Ses services sont organisés au sein d'une Direction générale située à Ouaga 2000.

Vu l'importance de ses activités et le nombre croissant de ses pensionnés, la CARFO s'est dotée d'une cellule informatique afin de toujours satisfaire sa clientèle tout en étant efficace et prompte. Pour ce faire, la cellule informatique se doit d'être la plus opérante que possible, car elle joue un rôle prépondérant dans la bonne marche de l'établissement. Ayant pour principales missions de mener des études en vue d'une meilleure exploitation du service informatique, de développer des applications et des bases de données indispensables à l'exécution des tâches des agents de la CARFO, ..., la cellule informatique a en plus la lourde tâche de s'occuper du parc informatique ainsi que de la gestion du réseau.

Le parc informatique de la CARFO compte près d'une centaine d'ordinateurs tous connecté sur le réseau et est en pleine croissance. Ainsi, pour mieux gérer ce parc et être efficace dans les interventions pour la résolution de problèmes, la CI veut se doter d'un outil de supervision du réseau. C'est dans cette optique que nous avons eu pour thème de stage : « **Etude et proposition d'une solution de supervision du réseau informatique basé sur le logiciel libre nagios** ». Ainsi, notre étude a pour but de proposer une solution idoine en vue de l'exploitation du logiciel libre nagios.

Ce rapport de stage s'articulera autour de quatre chapitres. Dans le chapitre un (1) nous présenterons la structure d'accueil et l'architecture du réseau informatique qui est en place. Nous ferons également une introduction à la supervision dans le chapitre deux (2) pour appréhender le concept de la supervision afin de mieux traiter notre thème. Une étude de nagios sera faite dans le chapitre trois (3), enfin au chapitre quatre (4), nous proposerons une solution de supervision du réseau avec une présentation de la mise en œuvre.

Chapitre 1 : Présentation de la CARFO et de son réseau informatique.

Introduction

Initialement connue comme un établissement public à caractère industriel et commercial, la CARFO est depuis le 3 avril 2008 un Etablissement Public de Prévoyance Sociale (EPPS). Ayant pour slogan « **bâtir une solidarité agissante entre les générations** », la CARFO est placée sous la tutelle de trois ministères qui sont :

- le ministère de la fonction publique et la reforme de l'Etat qui est chargé de la tutelle technique
- le ministère de l'économie et des finances chargé de la tutelle financière
- le ministère du commerce chargé de la tutelle de gestion.

Après avoir fait l'historique de la CARFO, nous présenterons aussi dans ce chapitre ses missions et attributions, ses orientations stratégiques, son organigramme et son réseau informatique.

1.1Historique

Avant les indépendances il n'existait pas de régime de retraite, ni de caisse de retraite. La gestion des pensions de retraite était confiée à une section de la direction du budget, qui avait pour mission d'assurer le traitement des dossiers des pensions constitués par les fonctionnaires, qui étaient affiliés à deux organismes de retraite différents. Le premier qui est la Caisse Locale de Retraite (CLR) avec son siège à Dakar, gérait le régime de retraite des anciens cadres locaux de l'ex Afrique Occidentale Française (A.O.F). Le second, appelé Caisse de Retraite de la France d'Outre Mer (CRFOM) basé à Paris était chargé de gérer le régime de retraite des fonctionnaires des anciens cadres généraux et supérieurs des ex Territoires d'Outre Mer (TOM).

Avec l'avènement des indépendances, ces caisses furent dissoutes en 1959 pour laisser la liberté à chacun des nouveaux Etats indépendants l'initiative de la mise en place de son propre système de retraite et d'en assurer la gestion. Notre pays, le Burkina Faso, à l'instar des autres ex-colonies a procédé à la mise en place progressive d'un dispositif juridique et organisationnel permettant la prise en charge de la retraite des agents de l'Etat. Il créa un

service des pensions dont l'organisation et le fonctionnement ont été définis par l'arrêté n°271/MF/B/DIR/F8 du 13/10/61.

Suite au manque d'expérience et de problème de gestion, l'Etat a jugé nécessaire de créer une structure autonome pour gérer les pensions, d'où la naissance de la CARFO en 1986. Ayant pour forme juridique d'un Etablissement public à caractère industriel et commercial dotée de la personnalité juridique et de l'autonomie financière, la CARFO a été transformée en un établissement public de prévoyance sociale (EPPS) par décret n°2008-155/PRES/PM/MFPRE/MEF du 3 avril 2008. Elle est régie par la loi n°016-2006/AN du 16 mai 2006 portant création de la catégorie d'Etablissements publics de prévoyance sociale et le décret n°2008-156/PRES/PM/MFPRE/MF du 3 avril 2008 approuvant ses statuts particuliers. La CARFO n'a débuté ses activités de façon autonome qu'en 1989.L'historique de la CARFO est relaté en intégralité sur son site officiel qui est www.carfo.bf.

1.2Missions et attributions

La CARFO a pour mission la gestion :

- du régime de retraite des fonctionnaires, militaires et magistrats institué par la loi n°47/94/ADP du 29 novembre 1994 portant régime général de retraite des fonctionnaires, militaires et magistrats étendu aux agents contractuels de la fonction publique recrutés à partir du 1er janvier 1999 à travers la loi n°006-2001/AN du 17 mai 2001 ;
- du régime de prévention et de réparation des risques professionnels institué par la loi n°022/2006/AN du 16 novembre 2006 portant régime et réparation des risques professionnels applicable aux agents de la fonction publique, aux militaires et aux magistrats ;
- de tout autre régime qui viendrait à être créé par la loi.

Se basant sur les missions qui lui sont assignées, la CARFO offre les prestations suivantes :

- La pension de retraite ;
- La pension d'invalidité ;
- La pension de survivants (veuves/veufs et orphelins) ;
- Les remboursements des retenues pour pension

La CARFO compte étendre son champ d'action en offrant très prochainement les prestations suivantes au titre des accidents de travail et des maladies professionnelles :

- Le paiement des rentes d'incapacité ;
- Le paiement des rentes de survivants ;
- La rééducation professionnelle, la réadaptation fonctionnelle ;
- La prise en charge du transport et des frais de séjour pour soins.

En plus de ses activités liées à son objet social, la CARFO assure aussi la liquidation et le payement des capitaux décès pour le compte de l'Etat.

1.3Orientations stratégiques

La CARFO a subie de nombreuses reformes notamment sa transformation en Etablissement Public de Prévoyance Sociale (EPPS), qui vise à faire d'elle une véritable institution de sécurité sociale autonome et déconcentrée, et en matière de protection sociale des agents de l'Administration publique une vitrine des pays de la sous régions. C'est dans cette optique que la CARFO s'est dotée en juillet 2008 d'un document d'orientation dénommé stratégie de renforcement du Système de Gestion de Retraite des Agents de l'Etat (SYGRAE). Le SYGRAE est un ensemble d'axes d'intervention et d'orientation des actions de la CARFO à l'horizon 2018. Il propose de relever trois défis majeurs qui sont :

- Réussir l'autonomie de gestion du système de sécurité sociale des agents de l'Administration Publique ;
- Assurer sa mutation vers les normes de la CIPRES ;
- Renforcer la confiance et l'espoir des usagers en elle.

Pour relever ses défis, des objectifs stratégiques ont été définis pour la décennie à venir qui sont notamment :

- Le renforcement des capacités de l'institution à faire face à ces obligations ;
- L'adaptation des prestations des services aux besoins de la clientèle ;
- Le renforcement de la viabilité financière du régime de retraite.

1.4 Organigramme de la CARFO

L'organigramme actuel de la CARFO est entré en vigueur en fin janvier 2008 et comprend une direction générale, des structures centrales et des structures déconcentrées.

- **La Direction Générale**

Assurée par un Directeur Général nommé par le Conseil d'Administration, elle comprend :

1. Le cabinet du Directeur général ;
2. le secrétariat général ;
3. le contrôle de gestion ;
4. le contrôle général ;
5. et la direction financière et comptable.

- **Les structures centrales**

Les structures centrales de la CARFO sont aux nombre de six (6) :

1. la Direction de l'Administration Générale (DAG) ;
2. la Direction du Recouvrement des Cotisations (DRC) ;
3. la Direction des Prestations (DP) ;
4. la Direction des Etudes et des Affaires Juridiques (DEAJ) ;
5. la Cellule Informatique (CI) ;
6. la Cellule des archives et de la documentation (CAD).

- **Les structures déconcentrées**

Les structures déconcentrées de la CARFO sont :

1. Les Directions régionales de la CARFO ;
2. Les agences de la CARFO.

Tous les textes portant organisation des directions et leurs attributions peuvent être consultés sur le site de la CARFO qui www.carfo.bf.

1.5 Le réseau informatique

Le réseau informatique de la CARFO fait parti du RESeaux Inter Administratif (RESINA) qui relie l'ensemble des administrations de Burkina Faso. RESINA est un réseau WAN avec une topologie en anneau FDDI (Fiber Distributed Data Interface), qui est étalé sur une dizaine de nœuds à Ouagadougou et relie les grandes villes du pays par des liaisons spécialisées (LS). Le débit de ce réseau dépend notamment des médias de communications et des équipements réseaux locaux utilisés.

RESINA couvre une large palette de besoin et offre les services de transport, d'administration, de sécurité, de partage d'information, d'accès aux informations et aux applicatifs, de communication et de travail coopératif. Il a pour avantage sa facilité d'emploi, son efficacité, sa sécurité et permet aussi de réduire les coûts engendrés par les charges.

Le réseau informatique de CARFO est reparti sur trois sites localisés dans la ville de Ouagadougou qui sont : le réseau de la carfo ouaga 2000, le réseau de la carfo paspanga et le réseau de la cellule des archives. L'ensemble de ces trois sites est interconnectés à travers RESINA, mis en place avec la technologie WiMAX (Woldwide Interoperability for Microwave Access) qui fait partie de la norme IEEE 802.16. Cette norme définie les transmissions de données à haut débit par voie hertzienne.

Sur chaque réseau on a :

- Un routeur pour interconnecter le réseau local au réseau étendu et à internet
- Des commutateurs (switchs) pour interconnecter les différents postes du réseau
- Des câbles à paires torsadés (UTP) de catégorie 6 qui partent des switchs vers les différents postes des utilisateurs.

La cellule informatique de la CARFO a établie sa propre politique de gestion de son parc informatique. En effet, tous les ordinateurs disposent d'un antivirus Kaspersky 6, l'attribution des adresses IP se fait automatiquement grâce à un DHCP (Dynamic Host Configuration Protocol) et chaque utilisateur possède un compte sur sa machine. Ce compte utilisateur est un compte limité qui n'offre pas les privilèges d'administration, mais seule, la cellule informatique dispose d'un compte administrateur « CARFO » sur chaque machine.
L'architecture du réseau peut être matérialisée à travers le schéma suivant.

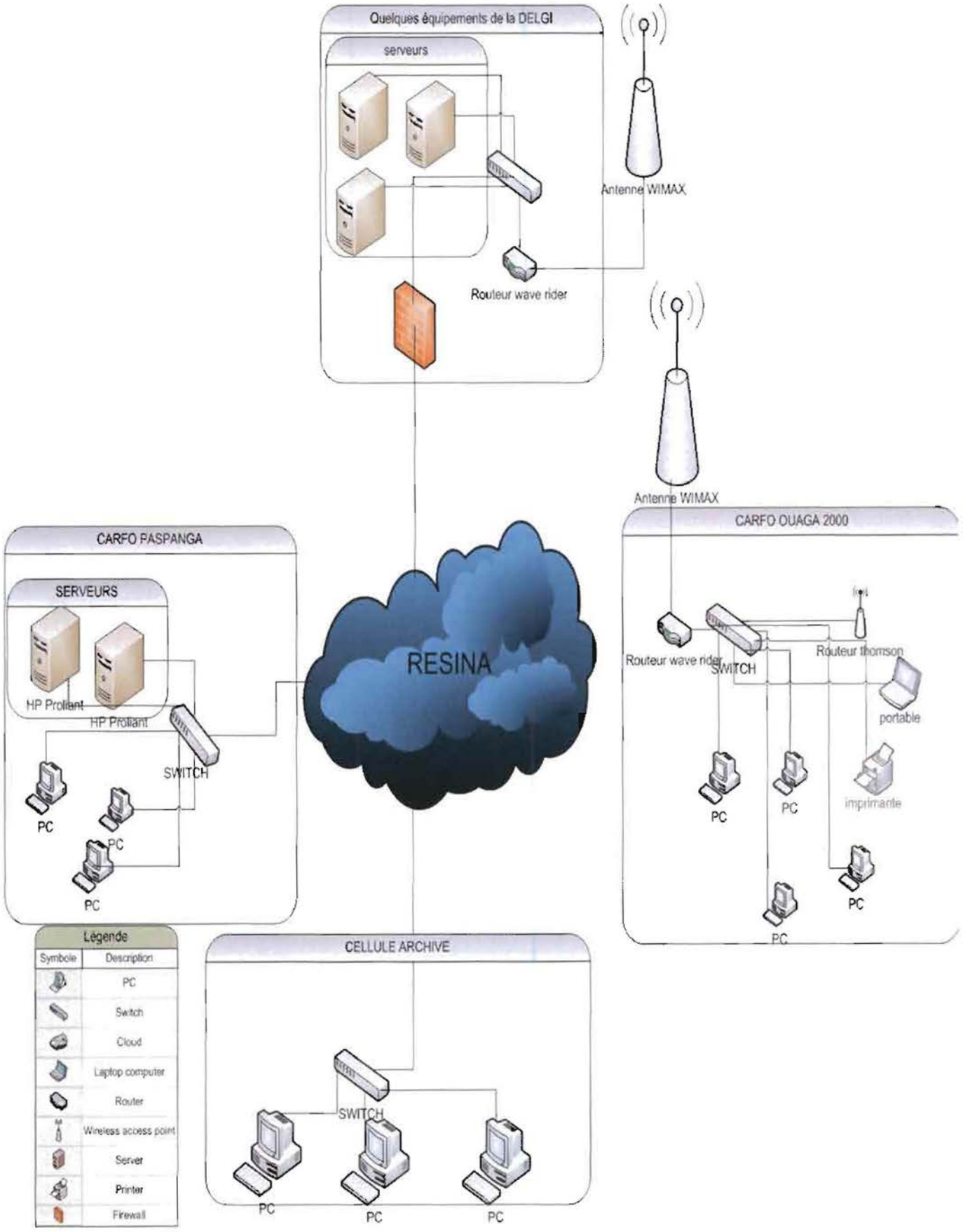


Figure1 : réseau de la CARFO

Conclusion

La CARFO est un établissement public de prévoyance social qui fait partie intégrante de la vie sociale de notre pays. Dans ce chapitre nous avons présenté l'historique de la CARFO, ses missions et attributions, ses objectifs et l'architecture physique de son réseau. Ayant une vision globale de son réseau, nous pouvons donc chercher à appréhender les concepts de la supervision informatique.

Chapitre 2 : La supervision informatique

Introduction

La gestion d'un système informatique est un travail permanent. Ainsi le service informatique ou l'administrateur réseaux doit savoir à tout moment l'état des équipements et des services sur les réseaux. C'est pour cela qu'on a recourt à une technique de suivi qui est la **supervision**, qui permet de surveiller, d'analyser, de rapporter et d'alerter les fonctionnements normaux et anormaux des systèmes informatiques.

Pour mener à bien leur mission de supervision, les administrateurs utilisent généralement des **moniteurs de supervision informatique**, qui permettent d'avoir une vue globale sur le fonctionnement du réseau, les performances des systèmes et d'alerter par différents moyens l'apparition d'une anomalie.

2.1 Objectifs et avantages

La supervision informatique à pour objectif principal d'aider les administrateurs réseaux pour la gestion quotidienne de leur réseau. Une politique de supervision informatique nous offre les avantages suivants :

- ✓ Une information précise sur l'état du réseau et des applications ;
- ✓ Une fiabilité et une crédibilité au service informatique ;
- ✓ Une réduction des pannes ;
- ✓ Augmentation de la sécurité ;
- ✓ Etc.

2.2 Méthodes de supervision informatique

Pour mettre en place une politique de supervision nous disposons de plusieurs méthodes :

- ✓ L'analyse des fichiers logs ou journaux : dans ces fichiers sont enregistrés l'historique du fonctionnement du système et des applications et analyser ces fichiers nous permet de savoir à quel moment une application à cesser de fonctionner.

- ✓ L'exécution et la récupération des résultats des commandes réseaux et des scripts locaux ou distantes :
- ✓ L'utilisation du protocole de gestion SNMP (Simple Network Management Protocol) : ce protocole permet la gestion des équipements et le diagnostique des problèmes du réseau.

2.3 Le protocole SNMP

SNMP (Simple Network Management Protocol) est à la fois un protocole applicatif, une collection de spécifications pour la gestion du réseau et un ensemble de définitions de structure de données avec des concepts associés. Il est né en 1988 de la nécessité de disposer d'un outil de supervision du réseau pour l'administration des réseaux étendus qui comportaient un grand nombre d'hôtes. Il est proposé par l'IETF (Internet Engineering Task Force) et offre les possibilités suivantes :

- ✓ Une cartographie du réseau ;
- ✓ Permet de mesurer en temps réel la consommation des ressources d'une application ;
- ✓ De signaler les dysfonctionnements ;
- ✓ Un ensemble de commandes pour le test des hôtes du réseau.

De nos jours SNMP demeure le protocole le plus repandu pour gérer les inters réseaux commerciaux, les réseaux universitaires et de recherche. Les fonctions du protocole SNMP sont simples et efficaces pour gérer les problèmes liés à l'administration des réseaux hétérogènes. Ainsi ce protocole offre les avantages suivants :

- ✓ C'est un protocole simple et facile à utiliser ;
- ✓ Permet la gestion centralisée d'un parc ;
- ✓ Dispose d'un modèle extensible ;
- ✓ Il est indépendant de l'architecture matérielle.

2.3.1 Historique

En 1987 il n'y avait que le protocole ICMP (Internet Control Message Protocol) qui était utilisé pour recevoir de l'information entre routeurs et hôtes. Ainsi, le couple de commandes **echo request** et **echo reply** était utilisé pour déterminer l'état des machines accessibles sur le réseau. En novembre de la même année, on assiste à la naissance du protocole SGMP (Simple

Gateway Monitoring Protocol avec pour RFC 1028) mais qui est trop orienté sur la gestion des routeurs. Il y a eu des tentatives de proposition de protocoles du côté de L'OSI avec le CMIS (Common Management Information Service) et le CMIP (Commom Management Information Protocol). Enfin, le protocole SNMP est sorti en 1988, comme une version améliorée du protocole SGMP et défini dans les RFCs 1155,1156 et 1157. La première version de SNMP qui est SNMPv1 a été standardisée en 1990, mais très rapidement les lacunes de cette version se font sentir notamment : le manque de hiérarchie, très peu de code d'erreur et de notifications, problèmes de sécurité. Suite aux lacunes de la version 1, la deuxième version voit le jour en 1993 et est nommée SNMPv2. Mais pour se faire une place dans le monde de la supervision, il faut que SNMPv2 se conforme aux nouvelles normes des éditeurs. Ainsi, on assiste au développement de sous versions de la version SNMPv2 qui sont :

- SNMPv2p : qui est une mise à jour de SNMPv1 qui corrige les failles de sécurité.
- SNMPv2c : appelé **community stringbased SNMPv2** qui améliore les opérations du protocole et des type d'opérations de SNMPv2p et utilise la sécurité par chaîne de caractères **community** de SNMPv1.
- SNMPv2u : dans cette version la sécurité est basée sur les usagers et utilise les opérations et les types de données de SNMPv2c.
- SNMPv2* : c'est une combinaison des meilleurs parties de SNMPv2p et SNMPv2u.

Parmi les sous version de SNMPv2, la version SNMPv2c était la plus utilisé, mais en 1997 la version 3 nommé SNMPv3 voit le jour et renforce la sécurité avec une gestion hiérarchisée de la SNMPv2x (x : sous version). Cette version est complexe et est difficile à implémenter avec une mise en œuvre délicate.

2.3.2 Fonctionnement

Le fonctionnement du protocole SNMP repose sur quatre (4) composantes à savoir : la station d'administration, l'agent de supervision, la base d'information et de management et le protocole de gestion du réseau.

- La station d'administration : c'est une station autonome communément appelée NMS (Network Management Station) qui comporte un ensemble de logiciels appelés applications d'administration réseau ou NMA (Network Management Application). La NMA est constituée d'une interface utilisateur qui permet aux administrateurs de gérer

le réseau. les administrateurs autorisés à se connecter à l'interface peuvent faire des requêtes aux agents, recevoir les réponses des agents, modifier les valeurs dans les agents et recevoir des évènements asynchrones des agents (trap). Généralement les NMS déclenchent des alertes en fonction des réponses obtenues des agents (alertes sonores et visuelles, pager, mail, SMS, ...). Dans le monde de la supervision, il existe beaucoup d'application d'administration que nous pouvons regrouper en deux (2) grands groupes à savoir : les applications d'administration propriétaires (Hewlett Packard OpenView, IBM Netview/600, Cisco Works, SunConnect SunNet Manager, etc.) et les applications d'administration libres (MRTG, Nagios, CACTI, Zabbix, Centreon, etc.).

- L'agent de supervision : ce sont des applications chargées de collecter les informations sur l'état de fonctionnement du matériel (hôtes, routeurs, switchs, passerelles) que l'on cherche à gérer. Ils ont accès aux données de management comme défini dans la MIB en lecture et en écriture, peuvent envoyer un évènement au manager (NMA) de façon asynchrone (trap), peuvent jouer le rôle de proxy pour un élément du réseau n'ayant pas le protocole SNMP implanté et peuvent déployer l'intégralité du protocole SNMP sur un hôte du réseau. l'agent peut être un daemon sur les systèmes UNIX/Linux ou un service sous Windows, ou faire partie intégrante d'un OS comme Cisco IOS, ou même implanté directement au niveau matériel comme (HP gCard, JetDirect). L'agent est simple à implémenter car il est stateless (ne stocke aucun état) et ne stocke aucune donnée.
- La base d'information de management : encore appelé MIB (Management Information Base) stocke les informations représentant les éléments du réseau et leurs attributs. Elle a un format standardisé qui est la SMI (Structure of Management Information défini dans la RFC 1065 et donnant pour chaque élément du réseau : un nom ou un identifiant numérique, une syntaxe (entier, string, char, ...), le type d'accès (read-only, read-write) et une description. Il existe beaucoup de MIB standardisés qui sont par exemple: MIB-II, ATM MIB (RFC 2515), BGB4 MIB (RFC 1657), Mail Monitoring MIB (RFC 2249), DNS Server MIB (1611), etc. Elles sont nombreuses car les constructeurs ont la liberté de créer leurs propres MIBs selon leurs besoins spécifiques. La MIB la plus utilisée est la MIB-II car elle est supportée par tous les agents SNMP et permet de définir toutes les informations réseau de base telles que les éléments génériques comme le nom de l'administrateur, la localisation géographique de l'équipement, la liste des interfaces réseau, la vitesse de transmission, le nombre

d'octets transmis. Etant donné que la structure du réseau est de plus en plus liée au système, il existe aussi une MIB spécifique au système d'exploitation qui gère la charge du CPU, l'utilisation des disques, le Nombre d'utilisateurs connectés, la table des processus, … appelé Host Ressources MIB défini par la RFC 2790. Tous les objets administrés sont organisés en une structure arborescente et hiérarchique et chaque objet administré représente une ressource, une activité ou une information à gérer qui est identifié par un identificateur d'objet unique ou OID (objects IDentifiers). Chaque OID aura par exemple les champs suivants :

- ✓ INTEGER : c'est un champ de 32 bits qui est utilisé pour décrire l'état d'un équipement.
- ✓ OCTET STRING : contient l'adresse MAC
- ✓ COUNTER : c'est un compteur qui représente un nombre sur 32 bits qu'on incrémente et qui repasse à zéro quand il dépasse 2^{32}-1
- ✓ OBJECT IDENTIFIER : champ pour décrire l'objet
- ✓ IpAddress : l'adresse IP de l'objet
- ✓ Gauge : c'est un nombre sur 32 bits qui définit la vitesse d'un port Ethernet
- ✓ Time Ticks : nombre sur 32 bits qui mesure le temps en centièmes de secondes
- ✓ Etc.

➢ Le protocole de gestion du réseau : SNMP est un protocole de gestion du réseau qui est basé sur le protocole UDP du modèle TCP/IP que nous avons étudié plus haut. L'administrateur du NMA envoie des requêtes SNMP à l'agent sur le port 161 pour déterminer l'état de l'hôte et reçoit les résultats des requêtes et des alertes sur le port 162. Les commandes basiques de SNMP sont

- ✓ GET : permet d'interroger une variable et en cas de validité de la commande la valeur correspondante à cette variable est retournée comme réponse à Get.
- ✓ GET-NEXT : la commande est utilisée à la suite de la commande get pour interroger la variable suivante.
- ✓ GET-BULK : est utilisé pour interroger plusieurs variables ou faire une requête sur une branche complète de l'arbre
- ✓ SET : permet à l'administrateur d'émettre une requête de modification sur un élément. Dans ce cas l'agent répond par un ACK.
- ✓ TRAP : c'est une requête émise par l'agent à destination du NMS pour signaler un fonctionnement anormal.

2.4 Les moniteurs de supervisions

De nos jours, superviser son réseau devient facile car le monde de la supervision déborde de logiciels. Ces logiciels qui existent sur le marché de la supervision sont fiables, sécuriser et permettent d'effectuer des vérifications sur les machines et les services sur le réseau local et distant, de collecter et vérifier les résultats des testes de supervision envoyés par les agents de supervision et d'alerter l'administrateur réseaux en cas de fonctionnement anormale d'un hôte ou d'un service. En se basant sur le coût d'achat de ces logiciels peut les classer en deux grands groupes qui sont : les logiciels payants et logiciels libres.

2.4.1 Les moniteurs de supervision payant

Les moniteurs de supervision payant sont proposés par les éditeurs de logiciel qui ont compris rapidement dès le début des réseaux que superviser sera la clé de réussite des systèmes informatiques et un atout pour les entreprises. C'est pour cela que les entreprises n'hésitent pas à investir pour l'obtention d'une solution de supervision. Alors vu la demande croissante, le nombre d'éditeurs augmente donnant naissance à une variété de logiciels et parmi eux ont peut citer :

- **HP OpenView :** c'est un logiciel destiné aux petites et moyennes entreprises et permet de centraliser les informations de supervision sur un seul poste. Il favorise la gestion du réseau en offrant aux administrateurs une vue globale du système d'information, un contrôle homogène des différents composants du système informatique et une vision des incidents et leur impact.
- **IBM Tivoli Monitoring :** c'est un logiciel conçu pour aider les entreprises à surveiller et gérer les matériels et les logiciels essentiels notamment les systèmes d'exploitation, les bases de données et les applications sur des environnements répartis. Il permet la surveillance de manière proactive des ressources systèmes vitales, détecte efficacement les goulets d'étranglement et les problèmes potentiels des applications et répond automatiquement aux évènements.

Ces logiciels possèdent tous les avantages dans le domaine de la supervision mais leur inconvénient face à la concurrence est leur coût d'achat élevé. Ainsi les entreprises ont tendance à se tourner vers le monde libre où les logiciels proposés commencent à être compétitifs et deviennent de plus en plus professionnels.

2.4.2 Les moniteurs de supervision libre

Les moniteurs de supervision libre sont des logiciels proposés gratuitement par des développeurs qui cherchent à se faire connaître à travers leurs produits en œuvrant dans le social. Parmi ces logiciels on peut citer :

- **Nagios** : c'est le moniteur de supervision le plus repandu parmi les logiciels open source et est suivi par toute une communauté de développeurs. Il permet la supervision du réseau et des systèmes et s'adapte aux systèmes d'information de moyenne taille au plus important. Nous allons l'étudier plus en détaille dans la suite de ce rapport.
- **MRTG** (Multi Router Traffic Grapher) : ce moniteur de supervision génère des pages HTML qui représentent en temps réel le trafic réseau. Son architecture logicielle permet l'intégration de composants hétérogènes, ainsi il s'intègre étroitement avec Nagios.
- **CACTI :** principal successeur de MRTG, il est basé sur RRDTool et permet de représenter graphiquement le statut des périphériques réseaux en utilisant le protocole SNMP ou grâce à des scripts écrit en perl, en C, ou en PHP. Le monitoring de CACTI est fondé sur la supervision réseau et la métrologie et cependant il est très efficace dans la gestion des données de performances.
- **ZABBIX :** c'est un moniteur qui est beaucoup plus orienté du côté de la supervision système. Il a une architecture tout-en-un avec des agents dédiés que l'on doit installer sur les éléments distants. Il est facile à installer et à configurer mais le revers de la médaille est qu'il ne gère pas les éléments imprévus que l'on souhaitera ajouter au plutard.
- **Zenoss :** c'est un concurrent direct des solutions de supervision payantes comme HP OpenView ou IBM Tivoli Monitoring car il est disponible en version commerciale avec des supports pour les administrateurs et version libre sous licence GPL. Il offre les fonctions de découverte, d'inventaire, de supervision de la disponibilité, gestion de la performance et des évènements.

Ces logiciels cités offrent tous l'avantage commun d'être gratuit, donc en fonction des besoins de supervision nous devons faire un choix parmi eux. Nous choisissons Nagios comme logiciel de base pour la supervision du réseau car c'est le plus repandu, dispose d'une forte communauté de développeurs, ses fonctionnalités couvrent tous les aspects de la supervision,

son architecture logicielle est modulaire et permet l'intégration d'autres logiciels comme CENTREON, CACTI, NAGVIS, … pour faciliter l'administration, le reporting et avoir une vue complète de tous les éléments du réseau.

Conclusion

Nous avons abordé la supervision informatique pour nous faire une idée de comment nous pouvons connaître l'état des hôtes et des services. Notre logiciel de base que nous allons utiliser est pour la supervision est nagios, qui est le plus rependu dans ce domaine et que nous présentons dans le chapitre suivant.

Chapitre 3 : Etude de Nagios

Introduction

Anciennement appelé **Netsaint**, **NAGIOS** est un moniteur de supervision permettant la surveillance active et passive de serveurs, équipements et services réseaux. Elle surveille les hôtes et services spécifiés, alertant lorsque les systèmes vont mal et quand ils vont mieux.

C'est un logiciel libre sous licence **GPL** reconnu dans le monde de la supervision et utilisé par les plus grands opérateurs. Un logiciel libre est un logiciel dont l'utilisation, l'étude, la modification, la duplication et la diffusion sont autorisées (techniquement et légalement). Les principaux acteurs de Nagios sont : Ethan Galstad pour le développement du daemon Nagios et les mises à jour des différentes versions et Karl Debisschop, Subhendu Ghosh, Ton Voon et Stanley Hopcroft pour le développement des plugins.

3.1 Historique

En 1999, Ethan Galstad développe et publie NetSaint comme moniteur de supervision libre et open source qui devient une alternative pour la mise en place d'une solution de supervision par les entreprises qui sont confrontés aux contraintes budgétaires et à la concurrence des offres des éditeurs de logiciels.

En 2002, pour des problèmes de propriété intellectuelle sur le nom, NetSaint devient Nagios et passe à sa version v1.

En 2004, on aura successivement la version v1.2 et la version v2

En 2009, on a les versions v3 et v3.2

En 2010, on a Nagios v3.2.3

3.2 Fonctionnalités

Nagios fonctionne sur linux et dans le cas général pour la plupart des systèmes Unix. Nagios offre les possibilités suivantes :

- La surveillance des services réseaux (SMTP, POP3, HTTP, NTP, PING, etc.)
- La surveillance les ressources des hôtes (charge du processeur, utilisation des disques, etc.)

- Le contrôle parallèle des services
- permet de définir la hiérarchie du réseau en utilisant des hôtes parents
- permet de contacter (par email, sms, alertes sonore, etc.) le(s) administrateur(s) en cas de problème ou pour signaler la résolution d'un problème
- La rotation automatique des fichiers journaux
- La capacité à définir des gestionnaires d'évènements permettant une résolution proactive des problèmes
- Permet la mise en œuvre d'une redondance des serveurs de supervision
- Une interface web permettant de voir l'état courant du réseau, l'historique des notifications et des problèmes, le fichier journal, etc.
- Etc.

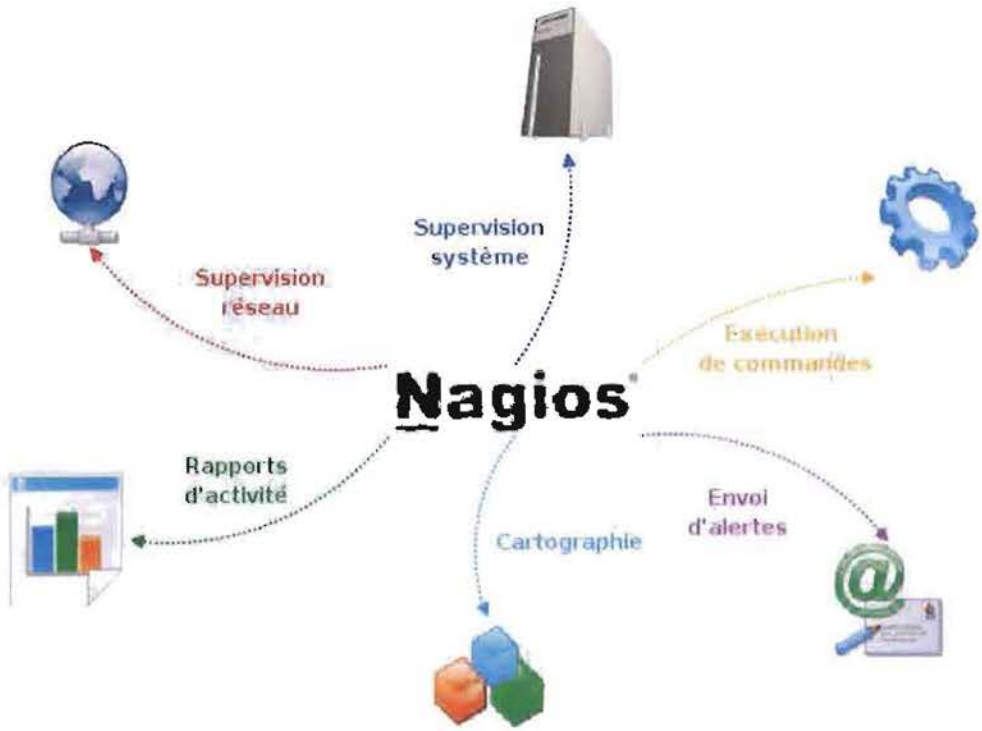


Figure5 : fonctionnalités de Nagios

3.3 Architecture

L'architecteur de Nagios peut se résumer à travers la figure ci-dessous

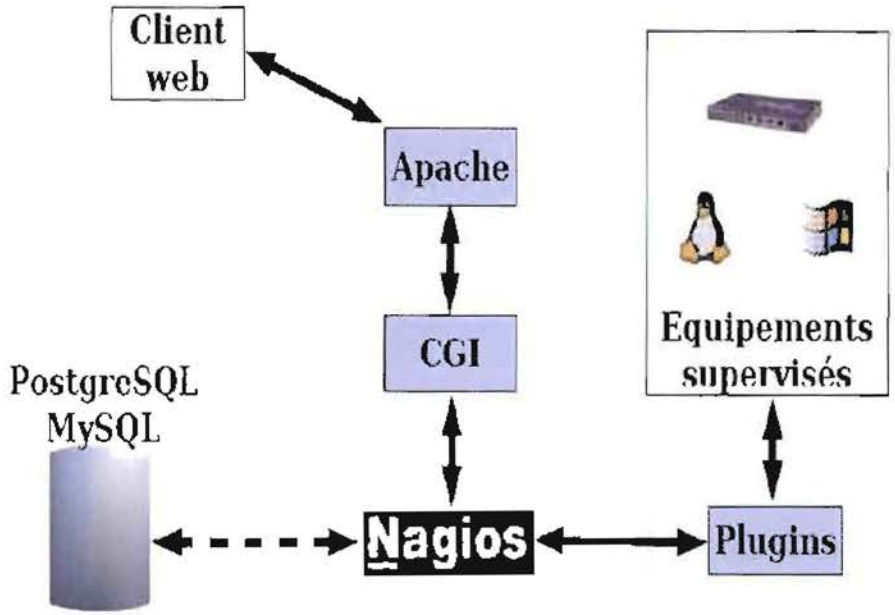


Figure6 : architecture de Nagios

Nagios est un programme modulaire composé de trois parties :

- ✓ Le moteur de l'application qui est un ordonnanceur qui gère :
  - L'ordonnancement et les dépendances des vérifications
  - Les actions à entreprendre quand un hôte ou un service change d'état (alertes, escalades, corrections automatiques).
- ✓ L'interface web, qui permet d'avoir une vue d'ensemble du système d'information à travers des CGIs. Les informations que présente l'interface web sont :
  - Une vue d'ensemble sur l'état de tous les hôtes et services supervisés à travers le cgi **status.cgi.**
  - Une carte de tous les hôtes définis dans le réseau grâce au cgi **statusmap.cgi**
  - Une interface WAP qui offre un accès aux informations sur l'état du réseau via un téléphone portable compatible internet grâce au cgi **statuswml.cgi**

- Un aperçu de tous les hôtes du réseau en utilisant une modélisation 3D dans un langage VRML offert par le cgi **statuswrl.cgi**
- Etc.

✓ Les plugins, qui sont des minis programmes que l'on peut compiler en fonction des besoins de supervision. Nagios exécute un plugin dès qu'il à besoin de connaître l'état d'un service ou d'un hôte et évalue le code de retour de ce plugin. L'avantage de cette architecture est qu'on peut contrôler tout ce que l'on veut à travers les plugins mais le revers est que nagios ne sait ce que l'on contrôle car il ne fait qu'interpréter les résultats.

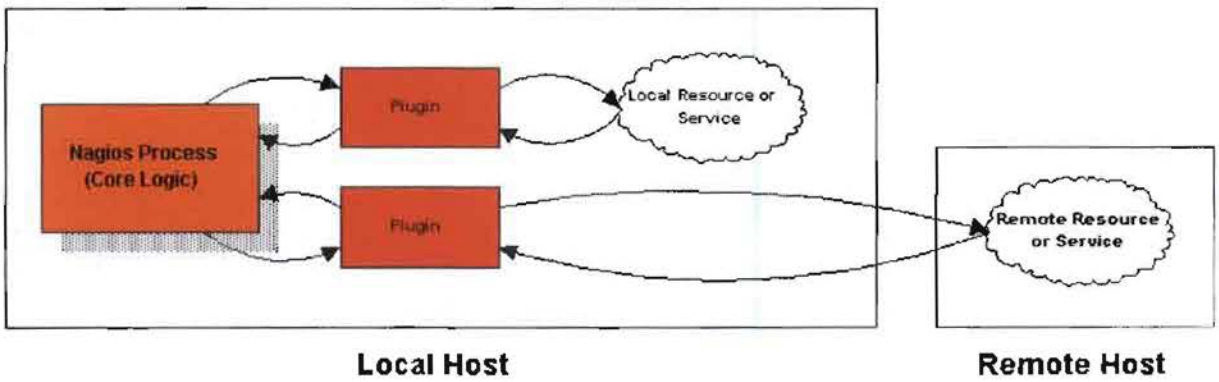


Figure 7 : fonctionnement des plugins

Il existe de nombreux plugins pour Nagios et nous allons présenter quelques uns dans ce rapport :

➢ Pour la supervision des ressources systèmes locales nous disposons des plugins suivants :

✓ **Check_load** : pour vérifier la charge moyenne (load average) sur les systèmes unix.

✓ **Check_swap** : pour vérifier l'état du remplissage du fichier d'échange

✓ **Check_mem** : pour vérifier l'état des espaces mémoire sur les systèmes unix

- ✓ **Check_disk** : pour vérifier les espaces disques disponibles sur les systèmes Unix.
- ✓ **Check_file_age :** pour vérifier qu'un fichier a été mis à jour et qu'il a une taille minimale.
- ✓ **Check_ntp_peer :** pour vérifier le temps du système par rapport au serveur de temps NTP.

➢ On a aussi des plugins pour superviser l'état physique de la machine

- ✓ **Check_ide_smart** : pour analyser des informations SMART des disques et détecter ceux qui sont défectueux
- ✓ **Check_sensors** : pour vérifier les informations que les systèmes unix exportent sur l'état de la machine.

➢ Il y a des plugins pour superviser les applications locales

- ✓ **Check_procs** : pour surveiller l'état des processus et leur priorité sur les systèmes unix
- ✓ **Check_mailq** : pour vérifier l'état des mails en attente d'envoi par sendmail ou mailq
- ✓ **Check_log2** : pour vérifier les nouvelles entrées dans un fichier log.

➢ Pour la supervision des services distants nous disposons des plugins suivants :

- ✓ **Check_tcp** : pour vérifier l'ouverture d'un port tcp
- ✓ **Check_udp** : pour vérifier l'ouverture d'un port udp
- ✓ **Check_dhcp** : pour vérifier qu'un serveur dhcp répond bien à la demande d'adresse IP
- ✓ **Check_dns** et **check_dig :** pour interroger un serveur dns et vérifier les enregistrements
- ✓ **Check_flexlm** : pour vérifier le bon état d'un serveur de licence de type flexm

- ✓ **Check_smtp** : pour vérifier le bon fonctionnement d'un serveur SMTP
- ✓ **Check_http** : pour permettre la vérification simple de l'accès à une page web
- ✓ **Check_oracle** : pour vérifier diverses informations sur l'état d'une base de données Oracle : connexion au listener, état des caches, remplissage des tablespaces, etc. ce plugin a besoin d'un client oracle pour fonctionner.
- ✓ **Check_mysql** et **check_pgsql :** pour tester des connexions à des bases de données MySQL et PostgreSQL.

➢ Pour la supervision des systèmes distants nous pouvons utiliser les plugins suivants :

- ✓ **Check_icmp, check_fping, check_ping :** pour verifier qu'une machine est connectée au réseau. Ces trois plugins effectuent ce test mais le plus léger est check_icmp.
- ✓ **Check_hpjd :** pour superviser les imprimantes à distance par SNMP
- ✓ **Check_centreon_snmp_traffic :** permet de suivre le trafic d'un élément réseau et vérifier qu'il est en bon état.
- ✓ **Check_snmp :** pour interroger une ou plusieurs OID et de comparer le résultat obtenu avec des seuils définis par l'administrateur.
- ✓ **Check_disk_smb :** pour vérifier l'état de partage des fichiers Windows et également vérifier que l'espace alloué n'est pas plein.
- ✓ **Check_ups :** pour vérifier l'état de fonctionnement des onduleurs de type UPS.
- ✓ **Check_snmp_load, check_snmp_mem, check_snmp_storage :** pour vérifier la charge du système, l'espace mémoire occupé, l'espace disque utilisé des serveurs distants via SNMP.
- ✓ **Check_snmp_win :** pour superviser l'état d'un ou de plusieurs services sous Windows.

➢ Nous avons des plugins utilitaires pour superviser l'état du processus Nagios

- ✓ **Check_nagios :** lorsque l'on met en place une architecture distribuée ou hautement disponible, ce plugin est nécessaire pour vérifier qu'au moins un des processus Nagios fonctionne toujours correctement.
- ✓ **Check_dummy :** pour vérifier l'état d'un élément dans le cas d'une supervision passive. Ce plugin renvoie comme résultat l'état de l'élément et le texte passé en argument.
- ✓ **Check_nrpe, check_by_ssh, check_nt :** pour vérifier le bon fonctionnement des agents nrpe, ssh et nsclient++, situés sur les hôtes.
- ✓ **Check_cluster :** pour faire une agrégation d'états au sein de nagios.

3.4 Les principes de base de Nagios

3.4.1 Détermination de l'état et du type d'état des hôtes

L'état courant des services et des hôtes est déterminé par deux composants : l'état du service ou de l'hôte (OK, WARNING, UP, DOWN) et le type d'état (HARD ou SOFT) dans lequel il se trouve.

3.4.1.1 Détermination de l'état des hôtes et/ou des services du réseau

Pour déterminer l'état de l'hôte ou du service, nagios évalue le code de retour des plugins. Nagios distingue quatre types d'états pour les services et trois pour les hôtes.

- ➢ Les états que peuvent avoir un hôte sont :
  - ✓ **UP :** il est en état de répondre ;
  - ✓ **DOWN** : indisponible ;
  - ✓ **UNREACHABLE**: état non connu.
- ➢ Les états des services sont :
  - ✓ **OK :** tout va bien
  - ✓ **WARNING :** quelque chose commence à mal aller
  - ✓ **CRITICAL** : la situation du service est très grave et nécessite une intervention immédiate.
  - ✓ **UNKNOWN** : la commande de vérification n'a pas pu obtenir les informations souhaitées.
- ➢ Le ci-dessous présente l'interprétation faite par Nagios des codes de retour des plugins

Code de retour	Etat de l'hôte	Etat du service	Couleur
0	UP	OK	
1	✓ UP : si la vérification forcée est activée. ✓ DOWN : sinon	WARNING	
2	DOWN	CRITICAL	
3	DOWN	UNKNOWN	

Les critères de test pour avoir ces états varient en fonction des plugins.

3.4.1.2 Détermination du type d'état des hôtes et/ou des services du réseau

Nagios compte deux types d'états : l'état soft et l'état hard. Ces états sont des éléments fondamentaux de la logique de supervision de nagios. Ce sont des outils d'aide de décision pour nagios qui déterminent quand il faut exécuter les gestionnaires d'évènements et envoyer les notifications.

- **L'état SOFT**

L'état SOFT signifie que le problème vient d'être détecté et demande une vérification. Il intervient dans les cas suivants :

- ✓ Une erreur SOFT : le résultat du contrôle d'un service ou d'un hôte est un état non-OK ou non-UP et quand le service n'a pas été contrôlé le nombre de fois spécifié par le paramètre **max_check_attempts** des définitions des hôtes ou des services.
- ✓ Un rétablissement SOFT : c'est quand un service ou un hôte se rétablit suite à un état d'erreur SOFT.

Lorsque des hôtes ou des services rencontrent des changements d'état SOFT il y a journalisation de l'état SOFT et les gestionnaires d'évènements sont exécutés pour le traitement de l'état SOFT.

- **L'état HARD**

L'état HARD signifie les états de bon fonctionnement (UP et OK) ou qu'un problème est confirmé. Il survient pour les hôtes et les services dans les cas suivant :

- ✓ Dans le cas d'erreur HARD : c'est quand le résultat du contrôle d'un hôte est non-OK et que le nombre de contrôle spécifié par l'option **max_check_attempts** de la définition d'hôte est atteint.
- ✓ Quand un hôte ou un service passe d'un état HARD à un autre état (par exemple de l'état WARNING à l'état CRITICAL).
- ✓ Quand le contrôle d'un service renvoie un état non-OK et que son hôte correspond est soit DOWN ou UNREACHABLE.
- ✓ Quand un service revient à un état OK après un état non-OK HARD. C'est un retour à la normale HARD.
- ✓ Quand un contrôle passif d'un hôte est reçu. Sauf si l'option **passive_host_checks_are_soft** est activée dans le cas contraire les contrôles passifs des hôtes sont traités comme un état HARD.

Lorsque l'état HARD intervient nagios journalise cet état, exécute les gestionnaires d'évènements pour le traitement de l'état et notifie les contacts d'un problème ou d'un retour à la normale d'un hôte ou d'un service.

3.4.2 Détermination de l'accessibilité des hôtes du réseau

Pour mener à bien sa mission de supervision, nagios a établit une politique de hiérarchisation des hôtes du réseau. Ainsi sur un même réseau, on a des hôtes locaux et des hôtes distants :

- ➢ **Les hôtes locaux**

Les hôtes locaux sont ceux qui se trouvent sur le même segment de réseau que l'hôte qui héberge le serveur nagios c'est-à-dire que aucun routeur ou firewall ne se trouve entre eux. Ainsi l'option « **parents** » de la définition d'hôte pour un hôte local doit être laissée vide, car les hôtes locaux n'ont pas de dépendance ou de parents. Le contrôle des hôtes sur le réseau local est très simple, il s'agit de savoir si l'hôte fonctionne ou pas et de déceler les problèmes de connexion réseau. Ainsi si nagios veut savoir si un hôte local fonctionne ou non, il lancera simplement la commande de contrôle de cet hôte. Si le résultat de la commande est un état OK, alors l'hôte fonctionne et dans le cas contraire l'hôte est hors fonction.

- ➢ **les hôtes distants**

Ceux sont les hôtes qui se trouvent sur un segment de réseau autre que celui de l'hôte qui héberge le serveur nagios. Ainsi dans un même réseau, certains hôtes peuvent avoir plusieurs

dépendances s'ils sont plus éloignés que d'autres. Cela permet de définir un arbre de dépendance des hôtes pour facilité la définition d'hôte de chaque hôte dans nagios. Ainsi l'option « **parents** » de la définition d'hôte d'un hôte distant doit être le nom de l'hôte directement au-dessus dans l'arbre de dépendance. Le contrôle des hôtes distants est plus complexe que celui des hôtes locaux. Ainsi pour contrôler un hôte distant nagios doit parcourir l'arbre de dépendance qui mène à cet hôte. Si le contrôle d'un hôte retourne un résultat non-OK nagios va parcourir l'arbre de dépendance jusqu'au sommet et déterminer si un problème sur un service résulte de l'arrêt de cet hôte, ou s'il s'agit de la rupture d'une liaison sur le réseau, ou simplement d'une erreur de service.

3.4.3 Le contrôle des hôtes et des services

Les hôtes et services sont contrôlés par le démon de nagios. Les contrôles sont effectués :

- A intervalles réguliers définit par les options **check_interval** et **retry_interval** dans les définitions d'hôtes et de services.
- A la demande quand un service ou un hôte change d'état, ou dans le cadre de la logique d'accessibilité de l'hôte via le réseau, pour le besoin de contrôle en prévision des dépendances des hôtes.

Ces principes de contrôles ainsi établies, on peut définir deux types contrôles pour les hôtes et les services qui sont : les contrôles actifs et les contrôles passifs.

3.4.3.1 Les contrôles actifs

Les contrôles actifs sont initiés par le processus nagios et exécutés à intervalles réguliers définis par les options **check_interval** et **retry_interval** dans les définitions d'hôtes et de services ou à la demande quand nagios a besoin de connaître le dernier état d'un hôte ou d'un service. Dans la pratique, quand nagios veut connaître l'état d'un hôte ou d'un service il procède de la façon suivante :

- ✓ Il va exécuter un plugin et lui passé les paramètres de contrôle dont il a besoin.
- ✓ Le plugin effectue le contrôle de l'état opérationnel de l'hôte ou du service et renvoie les résultats au processus nagios
- ✓ Le processus Nagios traite les résultats du contrôle de l'hôte ou du service et exécute les actions appropriées (lancer les gestionnaire d'évènements, envoyer des notifications, etc.).

3.4.3.2 Les contrôles passifs

Les contrôles passifs sont lancés et exécutés par des applications externes et soumis à Nagios pour traitement. Ils sont utilisés pour l'installation d'une supervision distribuée ou redondante, pour la supervision des services qui sont asynchrones par nature et ne peuvent pas être contrôlés activement de manière fiable (par exemples pour le traitement des interruptions SNMP, les alertes de sécurité, etc.). Les contrôles passifs fonctionnent de la manière suivante :

- ✓ une application externe contrôle l'état d'un hôte ou d'un service.
- ✓ L'application externe écrit le résultat du contrôle dans le fichier des commandes externes.
- ✓ Ensuite, nagios lit ce fichier et place les résultats dans une queue pour le traitement. Cette queue stocke les résultats des contrôles actifs et passifs.
- ✓ Nagios va exécuter périodiquement un évènement collecteur de résultat (fréquence de consolidation des résultats de contrôle « **check_result_reaper_frequency** ») et scanner la queue. Les résultats des contrôles actifs et passifs sont traités de la même manière et nagios va envoyer les notifications, les alertes de log, ..., en fonction du résultat de traitement des résultats de contrôles.

Pour autoriser les contrôles passifs il suffit juste de mettre à 1 dans le fichier de configuration principale l'option **accept_passive_service_checks** et aussi dans les définitions d'hôtes et de services l'option **passive_checks_enabled.**

3.4.4 Les notifications

Les décisions d'émettre des notifications sont prises dans le cadre du contrôle des services et des hôtes et ont lieu lorsque l'hôte ou le service change d'état ou quand il demeure dans un état HARD et non-OK, et que le délai défini dans le paramètre **notification_interval** de la définition d'hôte ou du service est écoulé depuis que la dernière notification à été émise pour ce même hôte ou service. Les notifications sont envoyées au groupe de contact défini par le paramètre **contact_groups** dans la définition d'hôte et de service. Avant l'émission, la notification passe par plusieurs étapes de contrôles appelées filtres qui jugent de la nécessité de notifier le contact ou pas. Les filtres à traverser sont :

- ➢ **Le filtre global au programme**

Ce filtre vérifie si le paramètre **enable_notification** est activé dans le fichier de configuration principal. Ce paramètre peut être modifié à tout moment en cours d'exécution via interface web.

- **le filtre d'hôte et de service**

  Ce filtre doit passer par cinq niveaux de vérifications avant d'atteindre le filtre suivant

  - ✓ ce filtre vérifie au premier niveau si l'hôte ou le service n'est pas dans une période d'un arrêt planifié : **scheduled downtime**. Si c'est le cas aucune notification n'est envoyée.
  - ✓ Au deuxième niveau il vérifie si l'hôte ou le service oscille. Si oui, aucune notification n'est envoyée.
  - ✓ Le troisième niveau de vérification lui permet de savoir si des notifications sont envoyées pour les états WARNING, CRITICAL, et RECOVERY pour les services et les états DOWN, UNREACHABLE, ou RECOVERY pour les hôtes.
  - ✓ Le quatrième niveau consiste à vérifier si la période définie par le paramètre **notification_period** est valide.
  - ✓ Le cinquième niveau consiste à vérifier si une notification avait été envoyé pour le même problème ou que l'intervalle de temps entre de deux notifications est atteint.

- **Le filtre de contact**

  Chaque contact spécifié par le paramètre **contact_groups** dans la définition d'hôte et de service possède son propre filtre de contact. Ce filtre possède deux niveaux de vérifications :

  - ✓ Le premier niveau pour vérifier pour chaque contact si des notifications doivent être envoyées quand les services sont dans les états WARNING, CRITICAL, et RECOVERY, et quand les hôtes sont dans un état DOWN, UNREACHABLE, ou RECOVERY.
  - ✓ Le deuxième niveau pour vérifier si la période de notification définie dans le paramètre **notification_period** dans la définition du contact est valide.

Les moyens que l'on peut utiliser pour notifier un contact sont : l'email, le pager, les alertes sonores, le téléphone pour l'envoi des sms, la messagerie instantanée Yahoo, une décharge électrique, etc. la méthode choisie pour notifier un contact est défini par la commande de

notification spécifiée dans la définition de contact. Les moyens de notification sont offerts par des applications externes à nagios.

3.4.5 Les périodes de temps

Les périodes de temps sont utilisées pour spécifier à nagios quand il doit superviser les éléments et quand il doit avertir les utilisateurs. Les périodes de temps varient en fonction des environnements et il convient à l'utilisateur de les définir librement. Les périodes de temps sont définies par le paramètre **check_period** dans les définitions des hôtes et des services et par le paramètre **notification_period** dans la définition du contact à notifier et a pour valeur par défaut **24x7** qui signifie que les contrôles des hôtes et services sont effectués par intervalle régulier à tout moment de la journée et pendant toute la semaine ainsi que pour l'envoie des notifications. Cette valeur par défaut est conseillée car définir ses propres périodes de temps signifie que pendant la période de non-contrôle :

- ✓ Le statut de l'hôte ou du service apparaîtra inchangé.
- ✓ Les contacts ne seront pas ré-notifiés des problèmes des hôtes ou des services.
- ✓ Si un hôte ou un service redevient OK, les contacts ne seront pas immédiatement notifiés du retour à la normale.

3.6 Installation

Nous disposons de trois méthodes pour installer nagios sur nos machines linux que nous allons décrire dans cette section :

- ➢ On peut utiliser le paquet précompilé **nagios3** en ouvrant le terminal et en tapant la commande suivante :

 # apt-get install nagios3 puis valider la commande avec la touche entrée

A la suite de cette commande les paquets suivant seront installés automatiquement : nagios3, nagios-plugins-basic, nagios-plugins-standard, libnet-snmp-perl, postfix, nagios-images, nagios3-common, radiusclient1, snmp, libradius1, nagios3-doc, bsd-mailx, nagios-plugins.

- ➢ La deuxième possibilité qui s'offre à nous, nous permet de télécharger les codes sources sur le site officiel de nagios, les compiles ensuite et pour les installés enfin.

Pour installer nagios correctement en suivant cette méthode, il faut disposer d'un compilateur C (GCC) avec toutes les bibliothèques associées, d'une librairie GD (requis pour les scripts CGI) et d'un serveur web (apache2).

- La troisième possibilité que nous disposons est l'exécution d'un script qui automatise l'installation de nagios. Ce script est proposé par nicolargo que l'on peut télécharger sur son site web à l'adresse suivante :

 # wget http://svn.nicolargo.com/nagiosautoinstall/trunk/nagiosautoupdate-ubuntu.sh

#chmod a+x nagiosautoupdate-ubuntu.sh

#sudo ./nagiosautoupdate-ubuntu.sh

Si vous installez correctement nagios, vous pourrez donc vous connectez à l'interface web, en lançant le navigateur web (Firefox) et dans la partie url vous tapez : soit http://localhost/nagios ou http://<adresse_ip_du_serveur>/nagios pour avoir accès aux interfaces de connexions.

3.7 Configuration

La configuration de nagios passe par l'édition et le paramétrage de ses fichiers de configuration. A la fin de l'installation de nagios, tous ses fichiers de configurations sont situés dans le répertoire **/usr/local/nagios** et des exemples de fichiers de configuration se trouve dans le répertoire **/usr/local/nagios/etc**. les principaux fichiers de configuration de nagios sont : le fichier de configuration principal, le fichier de configuration des ressources, le fichier de configuration des CGIs et le fichier de définition des objets.

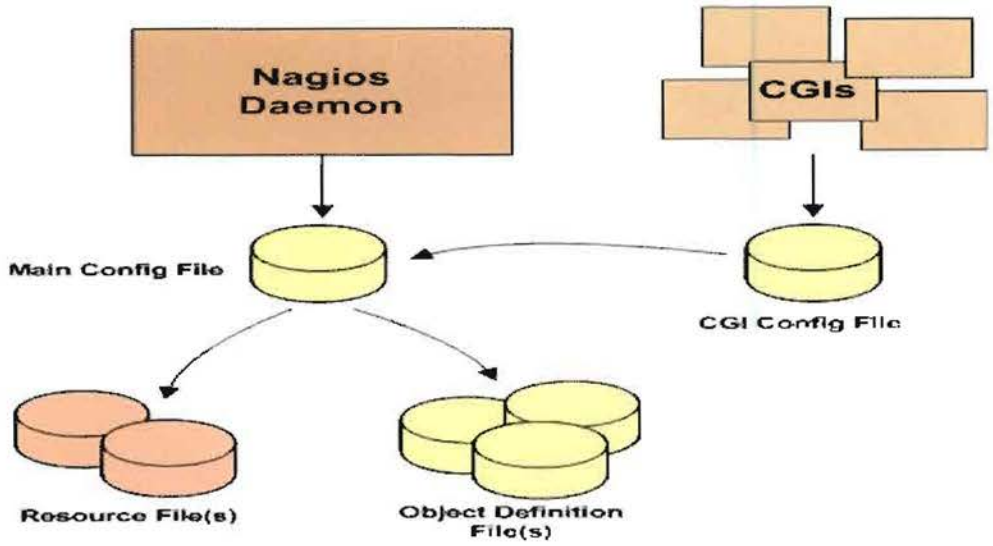


Figure 9 : configuration de Nagios

3.7.1 Le fichier de configuration principal

Le fichier de configuration principal ou main config file est situé dans le répertoire **/usr/local/nagios/etc** sous le nom de **nagios.cfg**. Les paramètres de ce fichier de configuration qui sont au nombre de 122 affectent directement le fonctionnement de nagios. Ce fichier est régulièrement consulté par le daemon nagios et par les CGIs.

3.7.2 Le fichier de configuration des ressources

Les fichiers de configuration des ressources ou resource file sont utilisés pour stocker des macros, des fichiers et des données sensibles de configuration. L'avantage que ces fichiers offrent est que les données de ces fichiers ne seront pas accessibles à travers les CGIs.

3.7.3 Le fichier de définition des objets

Les fichiers de définition d'objets ou object definition file sont utilisés pour définir les éléments du réseau que l'on souhaite superviser. Ainsi, on pourra définir dans ces fichiers les hôtes, les services, les groupes d'hôtes, les contacts, les groupes de contacts et les commandes, les périodes de temps, les Escalades de notifications et les dépendances d'exécution et de notification. Des exemples de fichiers de configuration se trouvent dans le fichier **/usr/local/nagios/etc/objects**. Nos fichiers de configuration de définition d'objets

doivent être déclarés dans le fichier de configuration principal avec les paramètres **cfg_file** et/ou **cfg_dir** pour permettre à nagios de les prendre en compte dans la supervision.

3.7.4 Le fichier de configuration des CGIs

Le fichier de configuration des CGIs ou CGI config file contient les paramètres (nom, mot de passe, autorisation aux différents CGIs, etc.) de l'administrateur nagios ainsi qu'une référence du fichier de configuration principal pour permettre aux CGIs de savoir comment est configuration nagios et où se trouve les fichiers de configuration des objects supervisés.

3.8 La supervision des éléments du réseau

Nagios permet la surveillance des machines fonctionnant avec des systèmes d'exploitation Windows, linux/Unix, des équipements réseaux (routeurs, Switch) et des imprimantes.

3.8.1 Supervision des machines Windows

La supervision des machines Windows permet de savoir la charge du processeur, l'espace mémoire utilisé, l'utilisation du disque dure, les utilisateurs connectés, les processus en cours d'exécution. La supervision des attributs et des services sur une machine Windows nécessite l'installation d'un agent sur celle-ci. Cet agent agit comme un proxy entre les plugins Nagios qui font la supervision des services ou des attributs sur la machine Windows. Sans installation d'agent sur la machine Windows, nagios serait incapable de superviser le moindre attribut ou service de la machine Windows. L'agent utilisé pour la supervision des machines Windows est NSClient++ avec le plugin check_nt installé sur le serveur Nagios.

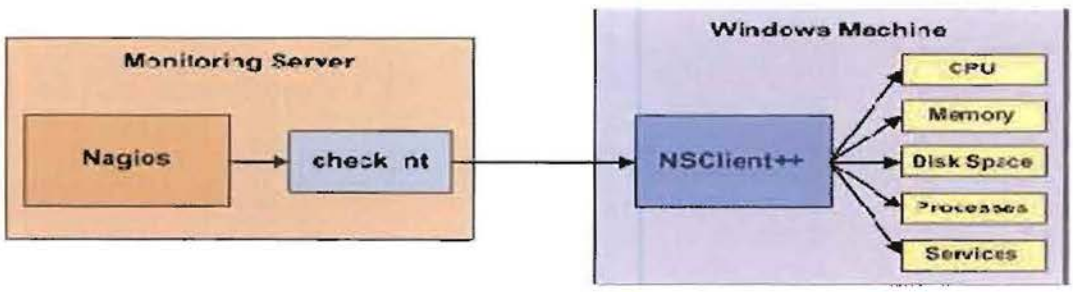


Figure 10 : supervision de machines Windows

3.8.2 Supervision des machines Linux/Unix

La supervision des machines Linux, nous permet de savoir la charge de processeur, l'espace mémoire utilisé, l'utilisation du disque dure, les utilisateurs connectés, les processus en cours

d'exécution. La supervision des machines Linux/Unix nécessite l'utilisation de NRPE avec le plugin check_nrpe installé sur le serveur Nagios.

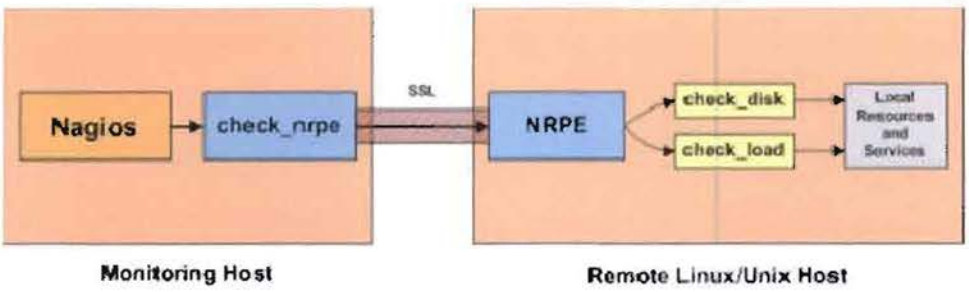


Figure11 : supervision des machines linux

3.8.3 Supervision des imprimantes réseau

La supervision des imprimantes permet de détecter les bourrages de papier, plus de papier, imprimante hors ligne, intervention requise, toner presque vide, mémoire insuffisante, porte ouverte, plateau de sortie papier plein etc. La supervision des imprimantes réseaux est très simple. Il suffit qu'elles soient compatibles avec la technologie JetDirect qui a en général le protocole SNMP activé, ce qui permet à Nagios de les superviser en utilisant le plugin check_hpjd.

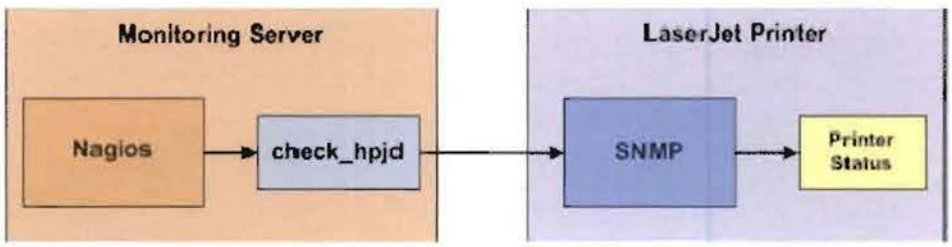


Figure 12 : supervision des imprimantes réseau

3.8.4 Supervision des routeurs et des switchs

Cette supervision permet de détecter les paquets perdus, d'avoir des informations sur l'état SNMP, de voir le traffic et la bande passante cosommée.la supervision se fait en utilisant les plugins check_snmp et check_mrtgtraf.

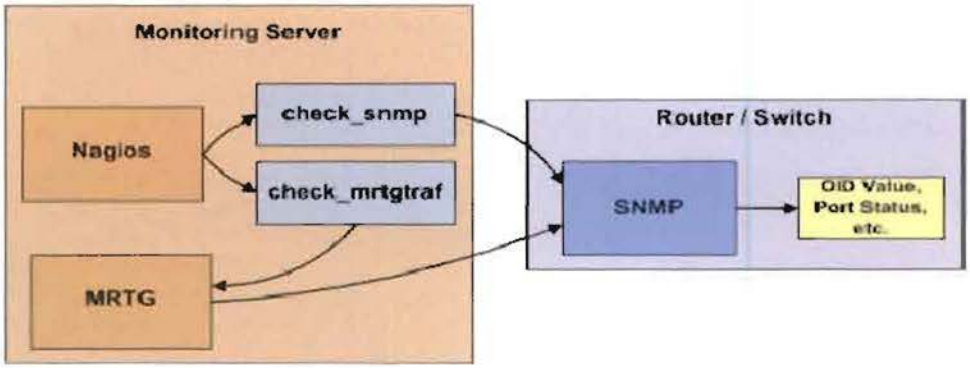


Figure13 : supervision des routeurs et des switchs

3.9 Les types de supervision

Nagios permet trois types de supervision qui sont : la supervision distribuée, la supervision redondante et la supervision haute disponibilité.

3.9.1 La supervision distribuée

La supervision distribuée consiste à utiliser plusieurs serveurs nagios pour la supervision. Nous aurons un serveur principal et des serveurs distribués. Le serveur principal aura pour fonctions d'exécuter les scripts de contrôles, de recevoir les résultats de contrôles des services et des hôtes et d'envoyer des notifications. Les serveurs distribués sont utilisés dans le but de décharger le serveur principal et ont pour rôle la supervision des groupes d'hôtes et des services associés à ces groupes. Ainsi, la configuration est différente entre le serveur principal et les serveurs distribués. Étant donné que les serveurs distribués supervisent seulement les groupes d'hôtes et leurs services on procède simplement à l'installation du daemon de nagios avec quelques plugins. On procède sur le serveur principal à l'installation complète de nagios (daemon, plugins, interface web,...). La communication est passive entre le serveur principal et les serveurs distribués car régulièrement les serveurs distribués rendent comptes de l'état des groupes d'hôtes et leurs services au serveur principal. Le schéma ci-dessous illustre l'architecture d'un réseau fonctionnant avec une supervision distribuée.

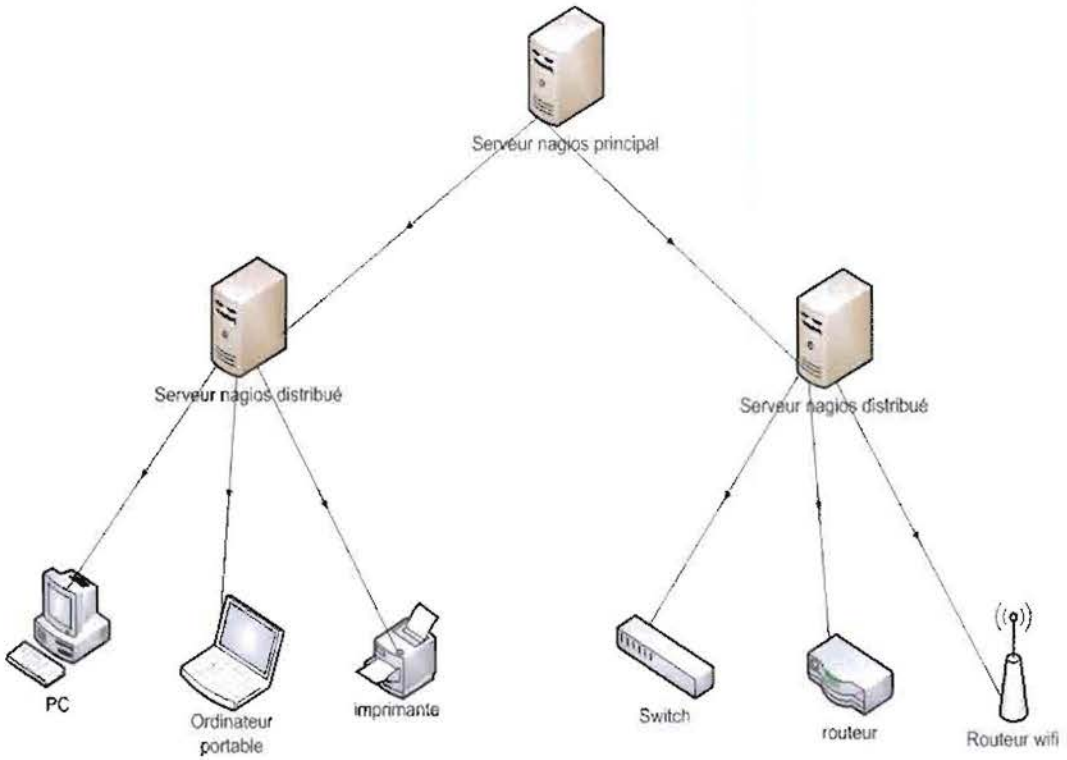


Figure 14 : architecture de la supervision distribuée

3.9.2 La supervision redondante

La supervision redondante utilise deux serveurs nagios pour la supervision des hôtes et des services réseaux. Le premier serveur appelé serveur maître envoie les notifications en cas de problème. Le second serveur appelé serveur esclave quant à lui supervise les hôtes et les services et n'envoie les notifications qu'en cas d'indisponibilité du serveur maître.

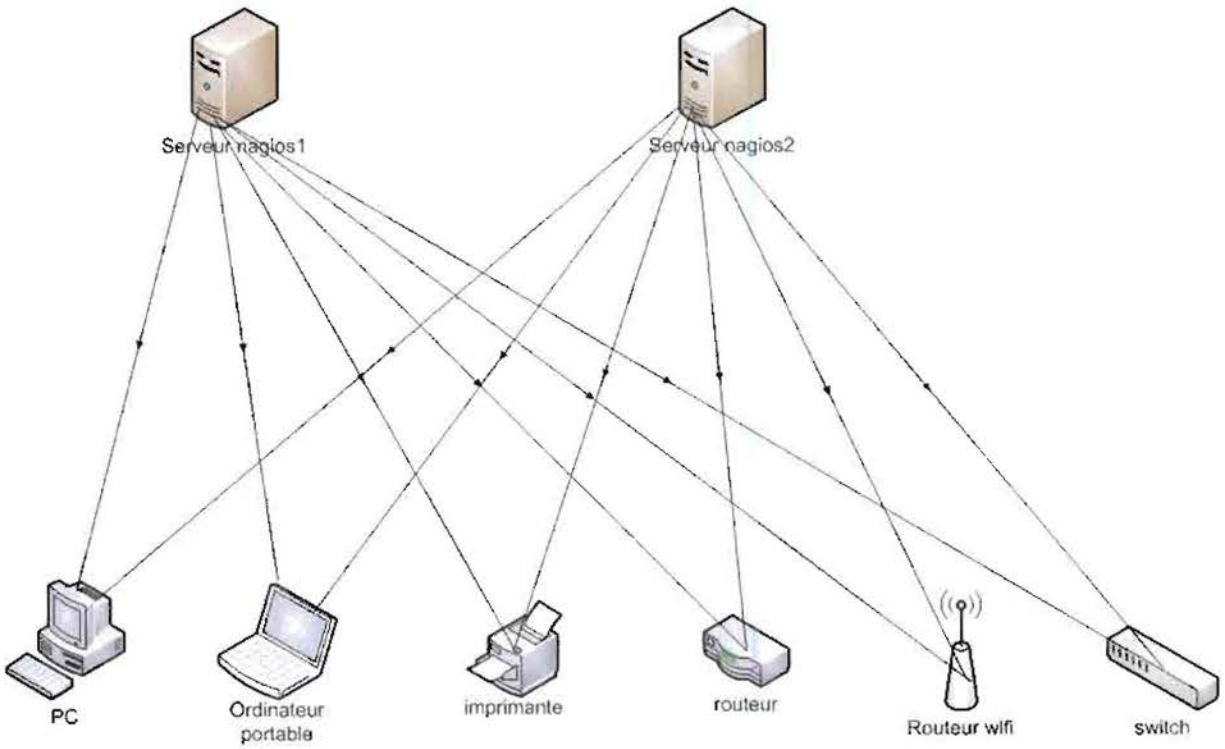


Figure 15 : architecture de la supervision redondante

3.9.3 La supervision haute disponibilité

La supervision haute disponibilité utilise aussi deux serveurs nagios. Un serveur maître qui surveille activement les hôtes et les services et un serveur esclave qui a les mêmes fichiers de configuration que le serveur maître. Le serveur esclave est chargé de surveiller le processus nagios du serveur maître et reçoit donc les résultats de contrôle des hôtes et des services du serveur maître. Il n'intervient qu'en cas d'indisponibilité du serveur maître pour le contrôle des hôtes et des services et aussi pour l'envoie des notifications.

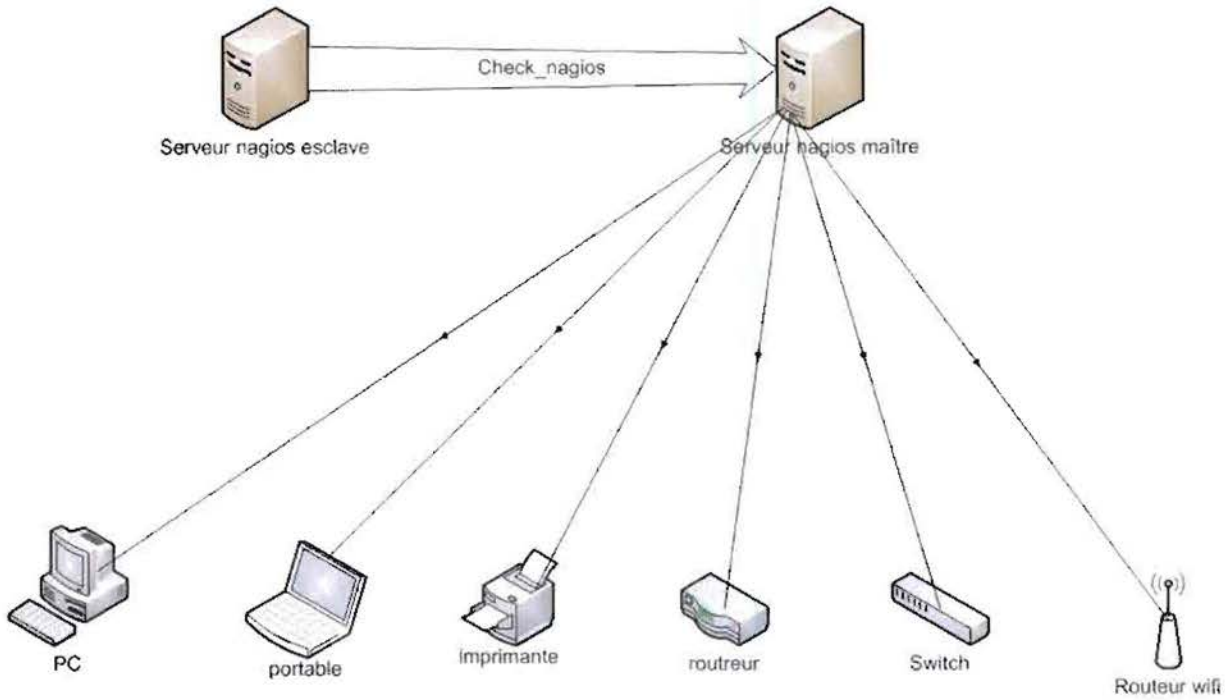


Figure 16 : architecture de la supervision haute disponibilité

Conclusion

Nagios est un moniteur de supervision qui a dix ans d'expérience et dont les possibilités couvrent largement nos besoins de supervision. Cette étude nous a permis de connaître son histoire et son évolution, ses possibilités, son architecture, ses principes de bases, sa configuration, les types de supervision et les moyens dont il dispose pour superviser les hôtes.

Nous pourrons donc nous baser sur cette étude pour la réalisation de notre projet de supervision du réseau informatique de la CARFO.

Chapitre 4 : proposition de la solution de supervision et mise en œuvre

Introduction

La réussite d'un projet de supervision est autant technique qu'organisationnelle, ainsi pour proposer une solution de supervision du système informatique nous suivrons les étapes suivantes :

- ✓ Un inventaire de tous les équipements que l'on doit superviser
- ✓ Sur chaque équipement actif nous déterminerons les objets à prendre en charge
- ✓ Nous regrouperons les objets en entité logique (groupe d'objet, groupe de service) pour la création et la bonne gestion des fichiers de configuration.

Ces étapes franchies, nous serons en mesure de proposer une solution de supervision de tous les équipements du réseau.

4.1 Inventaire des équipements du réseau

Le réseau informatique de la CARFO est essentiellement composé d'ordinateurs de Bureau, d'imprimantes, de serveurs, d'onduleurs et de routeurs. Le tableau ci-dessous résume les équipements à superviser :

Intitulé	Caractéristiques	Nombres	Observation
PC de Bureau	HP Compaq 2300 Microtower HP xw4400 workstation HP D330 DT DELL Vostro 220s SERIES	80	Pris en compte pour la supervision
imprimantes	Matricielle Tally T2280+ Tally Genicon 6312 line printer Lexmark E260dn Lexmark C734 dn HP LaserJet 4250	41	Seules les imprimantes Lexmark E260 dn et C734 dn ne seront pas supervisées
serveurs	HP Proliant ML530 HP Proliant ML350	1 1	Pris en compte pour la supervision
routeurs	WAVE RIDER NLC 1170	2	Supportent le protocole SNMP v1
Switchs	D-LINK DESS 1228	4	

4.2 Détermination des objets à prendre en compte pour la supervision

Pour une bonne supervision nous devons examiner les équipements composant le réseau et identifier les éléments à prendre en compte pour la supervision. Ainsi nous avons examiné les ordinateurs de bureau, les imprimantes, les serveurs, les routeurs et les onduleurs et déterminer les éléments à prendre en compte dans la supervision.

- Sur tous les PCs de Bureau et les serveurs, nous allons superviser l'utilisation des disques, de la mémoire, la charge du CPU, l'état de fonctionnement des applications (NSCLient, explorer, …).
- Sur les imprimantes nous allons superviser l'état de l'imprimante (pas de papier, bourrage, etc) et tester régulièrement la connectivité. Actuellement les imprimantes ne sont pas prises en compte dans le plan d'adressage du réseau et nous devons donc les intégrer en leur attribuant des adresses IP.
- Les routeurs sont supervisés pour déterminer le temps de fonction, l'état des interfaces, l'utilisation de la bande passante et du trafic réseau.

- Les onduleurs pour connaître leur état de fonctionnement.

4.3 Regroupement des objets en entités logique

Le regroupement des objets en entité logique, nous permettra de simplifier la configuration des hôtes et services en se basant sur des modèles de définition d'hôte, de service et de contact, d'établir une fiche de collecte d'information pour relever les noms de machine et leur adresse IP, et de créer un même modèle de fichier de configuration des ordinateurs de bureau et des serveurs.

4.3.1 Modèles

Pour la supervision des hôtes et des services, nous nous sommes basé sur les modèles de définition d'hôte, de services et contact suivant qui sont dans le fichier **/usr/local/nagios/etc/objetcts/templates.cfg.**

L'utilisation de ces modèles, nous offre l'avantage d'appliquer les mêmes commandes pour les hôtes et services afin de simplifier la tâche de supervision et d'alléger les fichiers de configurations. Ainsi, nous créerons des groupes d'hôtes à partir de ces modèles en fonction de la position géographie de chaque hôte qui sont : carfo2000r1a, carfo2000r1b, carfo2000r2a, carfo2000r2b, carfo2000r3a et carfo2000r3b pour les hôtes de la CARFO ouaga 2000, et carfopaspangaG1, carfopaspangaG2 et carfopaspangaG3 pour les hôtes de la CARFO de paspanga et enfin le groupe CAD pour les hôtes de la cellule des archives.

4.3.2 Fichier de configuration des serveurs et des PCs de bureau

Dans le fichier ci-dessous sont définis les services utilisés pour la supervision de PC de bureau et des serveurs

```
define host {
        use                     windows-server
        host_name                               #nom de l'ordinateur
        alias                   #nom de l'utilisateur
        parents                 #si l'hôte n'est pas sur le même LAN que le serveur
        address                 #adresse IP}
```

```
# definition de groupe
define hostgroup {
                hostgroup_name          #Nom du groupe

               Alias                    #service
}
#définition de la version du service NSClient
define service {
                use                     generic-service
                host_name               nom de l'ordinateur
                service_description             NSClient++-0.3.8-win32.msi
                check_command           check_nt!CLIENTVERSION
}
#service de surveillance des temps d'activité
define service {
            use                 generic-service
            host_name           nom de l'ordinateur

            service_description Uptime
            check_command               check_nt!UPTIME
}
#service de surveillance de la charge du processeur
define service {
            use                 generic-service
            host_name           nom de l'ordinateur
            service_description CPU Load
            check_command               check_nt!CPULOAD!-1 5,80,90
}
#surveillance de la charge du processeur
define service {
            use                 generic-service
            host_name           nom de l'ordinateur

            service_description Memory Usage
```

```
        check_command           check_nt!MEMUSE!-w 80 -c 90
}
#surveillance de l'espace disk
define service {


        use                     generic-service
        host_name               nom de l'ordinateur

        service_description     C:\Drive Space
        check_command           check_nt!USEDDISKSPACE!-1 c -w 80 -c 90
}
# surveillance de internet explorer.exe
define service {
        use                     generic-service
        host_name               nom de l'ordinateur
        service_description     Explorer
        check_command           check_nt!PROCSTATE!-d SHOWALL -1
Explorer.exe
}
```

4.4 Proposition de la solution de supervision

Mettre en place une solution de supervision n'est pas une tâche facile, car avant de superviser nous devons prendre en compte les éléments à superviser, analyser les systèmes utilisés et tenir compte des limites de nagios en termes de disponibilité et de performance. Face à cela nous proposons une solution de supervision de type haute disponibilité pour nagios qui va utiliser une base de donnée gérer par NDOUTILS et Centreon pour faciliter l'administration.

4.4.1 La supervision haute disponibilité pour nagios

Dans l'étude de nagios, nous avons vu qu'il offre trois types de supervision à savoir : la supervision redondante, distribuée et la haute disponibilité. Après analyse de ces trois types de supervision nous pouvons dire que :

- La supervision redondante offre l'avantage de faire une double supervision des hôtes et des services. Ce type de supervision consomme énormément la bande passante donc pose alors un problème de trafic réseau et de surcharge des systèmes supervisés.
- La supervision distribuée décharge le serveur principal mais n'est vraiment pas nécessaire quant on a principalement que des hôtes et des services à superviser.
- La supervision haute disponibilité peut être vue comme l'amélioration de la supervision redondante qui corrige le problème de trafic réseau et de surcharge des systèmes supervisés. Ce type de supervision utilise aussi un serveur maître et esclave, mais le serveur esclave n'agit qu'en cas d'indisponibilité de son maître.

La supervision haute disponibilité que nous avons choisie est aussi appelée architecture active/passive, car le nagios maître est actif pendant que nagios esclave est passif jusqu'à ce que le maître soit hors jeu. Le schéma ci-dessous illustre le fonctionnement de l'architecture actif/passif.

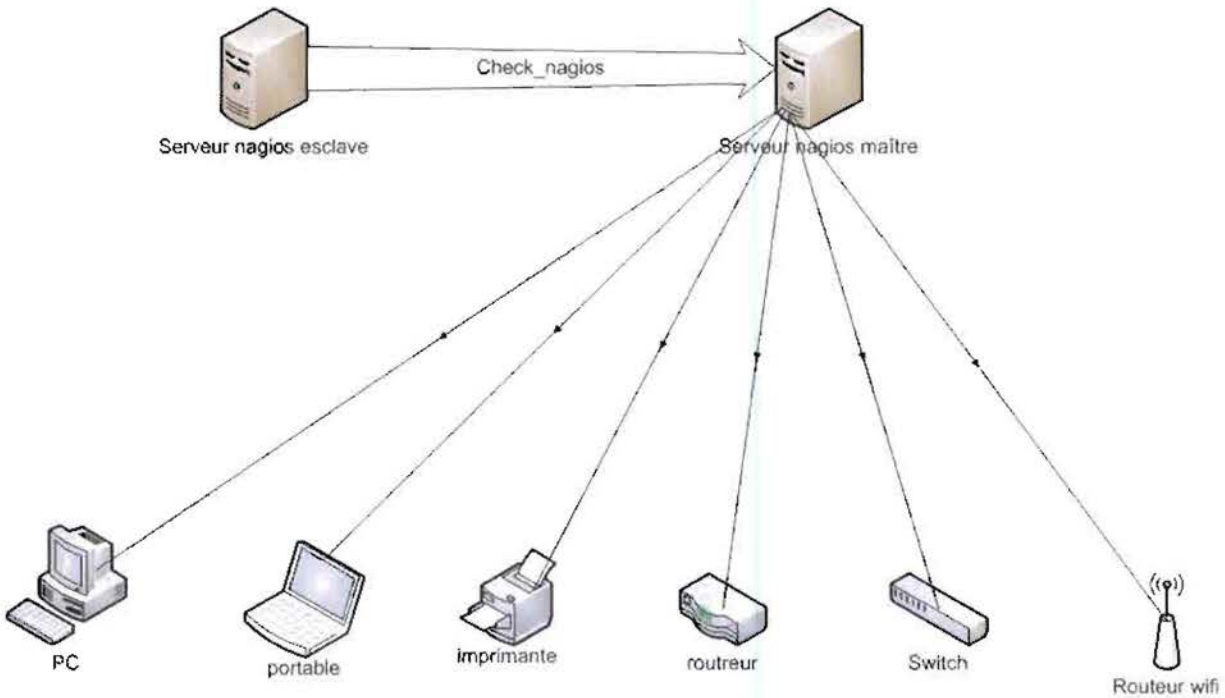


Figure17 : supervision haute disponibilité

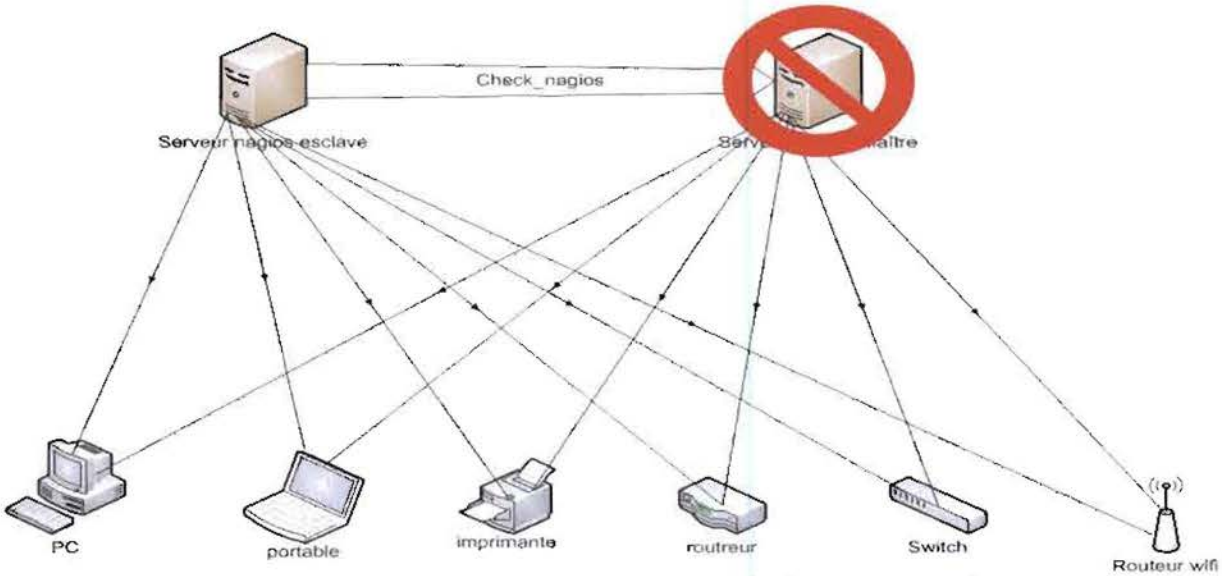


Figure 18 : réaction du serveur nagios esclave en cas d'arrêt du serveur nagios maître

L'enjeu majeur est de réaliser une communication en temps réel avec les deux serveurs. Cette communication permettra la réplication des fichiers de configuration, de savoir l'état précédent des hôtes et services supervisés. L'état précédents des hôtes et services supervisés est conservé dans le fichier de rétention d'état appelé **retention.dat**. Pour partager ce fichier, on peut simplement installer un système de fichier partagé NFS et le monté. Dans cette architecture, le nagios passif est en réalité un nagios semi-dormant qui ne lance pas les vérifications des hôtes et services mais attend que nagios actif lui envoie passivement les états. Nagios actif peut envoyer les états des hôtes et services à nagios passif par le biais de NSCA. Le schéma illustre le fonctionnement de l'envoie des états par le nagios actif au nagios passif.

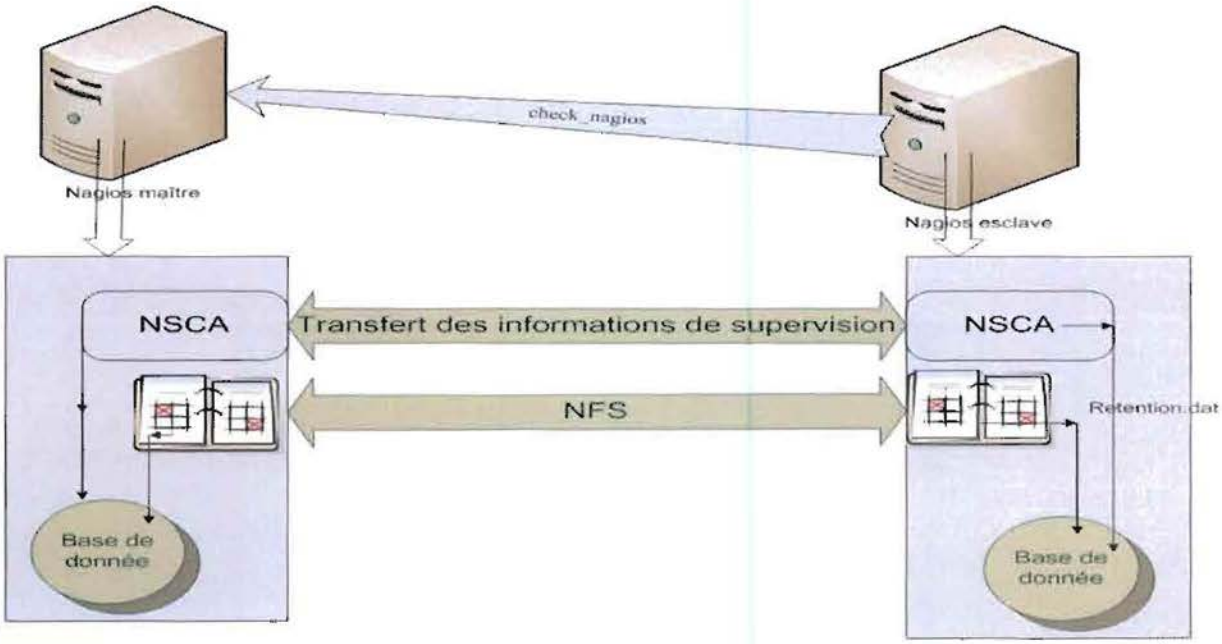


Figure19 : communication entre nagios actif et passif

4.4.2 Ndoutils

Ndoutils est conçu pour stocker toutes les données de configuration et d'évènements de nagios dans une base de données. Le stockage des informations de nagios dans une base de données permettra d'accélérer la récupération et le traitement des données et servira de base pour le développement d'interface web en PHP dans les nouvelles versions de nagios. Les bases de données actuellement pris en charge par Ndoutils sont MySQL et PostgreSQL. Il a été conçu pour fonctionner pour les utilisateurs qui ont une seule installation de nagios, des installations multiples et autonomes, pour les supervisions de type redondante, distribuée et haute disponibilité. Les données de chaque processus de nagios peuvent être stockées dans la même base de données ou dans différentes base de données. Ndoutils est principalement composé de quatre éléments qui sont : le module NDOMOD broker évènement, le LOG2NDO utility, le FILE2SOCK utility et le NDO2DB daemon. Parmi, les quatre éléments qui composent Ndoutils le module NDOMOD broker évènement et le daemon NDO2DB nous intéresse. Le module NDOMOD a été conçu pour l'exportation des données de configuration ainsi que des informations sur différents évènements d'exécution qui se produisent dans le processus de suivi du daemon nagios. NDO2DB, quant à lui est chargé de recevoir les données et de les placer dans la base de données.

4.4.2 Centreon

Centreon est un logiciel de supervision réseau, basé sur le moteur de récupération d'information nagios. Il fournit une interface web simplifiée pour rendre la consultation de l'état du système accessible à un plus grand nombre d'utilisateurs, notamment à l'aide de graphiques. Les fonctionnalités de Centreon sont citées ci-dessous :

- Une interface multiutilisateur intuitive et personnalisable
- Une interface de configuration évoluée pour configurer le périmètre à superviser
- Des aides à la configuration
- Une gestion de tous les fichiers de configuration de Nagios (*cgi*, *nagios.cfg*...)
- Un module de chargement de configuration de Nagios
- Une compatibilité Nagios 1.x, Nagios 2.x, Nagios 3.x
- Un test de validité des configurations avec le debugger de Nagios
- Des fiches d'identités serveurs/équipements réseau regroupant les informations de base sur ces types de ressource
- Des représentations graphiques élaborées et personnalisables
- Une gestion des accès très fine, comprenant les ressources comme les pages de l'interface
- Un système de modules qui permet l'inclusion d'autres applications dans Centreon
- Un compte-rendu complet sur les incidents
- Un système de calcul de la qualité de service en temps réel avec alerte en cas de diminution de la qualité de service

L'utilisation de Centreon dans notre solution nous permettra donc de configurer les hôtes et services via interface web, de lire et enregistrer les informations dans la base de données et enfin de générer automatiquement les fichiers de configuration de nagios. La figure ci-dessous décrit le fonctionnement du processus de configuration.

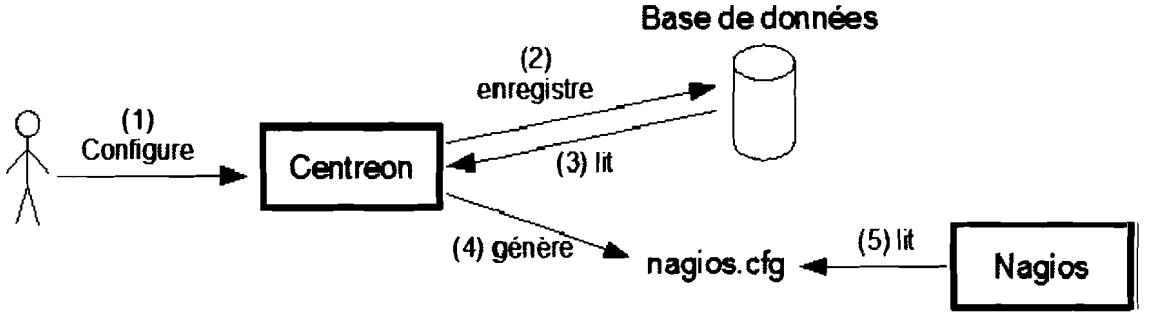


Figure 20 : processus de configuration

4.5 Les besoins

Pour la mise en œuvre de cette solution, nous avons fait l'évaluation des besoins matériels et logiciel et aussi l'évaluation des coûts du projet. Le tableau suivant fait l'état des besoins pour la mise en œuvre.

Désignation	**Caractéristiques**	**Quantité**	**Prix unitaire**	**Montant**
Ordinateur de bureau	HP Compaq DX2400 Core duo cpu: 2Ghz; RAM: 2Go; disque dur: 250Go	2	400000 Fcfa	800000 Fcfa
Système d'exploitation	Ubuntu server 10.4			
Nagios	Version 3.2.3			
Nagios-plugins	Version 1.4.15			
Ndoutils	Version 1.4b9			
centreon	Version 2.0.2			
postfix	Version 2.5.5-1.1			
NSCA	Version 2.7.2			
NRPE	Version 2.12			
NFS server	Version 1.1.1.2-2			
Mysql-server	Version 5			
Nsclient++-0.3.8.msi	Version 0.3.8			
Main d'oeuvre		1	800000 fcfa	800000 fcfa
Temps d'exécution			3 mois	
total			**1600000 fcfa**	

4.6 Mise en œuvre

4.6.1 Installation du système

Nous énumérons ici les grandes lignes de l'installation de ubuntu-server 10.4

1. Choix de la langue : FRANÇAIS
2. Origine du clavier : français
3. Disposition du clavier : latin-9
4. Nom de la machine : serversupervisionci
5. Configuration de l'horloge : afrique/ BURKINA FASO
6. Partitionner les disques : manuel
7. Identifiant pour le compte utilisateur : superviseurci
8. Mot de passe : mot_de_passe
9. Installation du grub
10. Fin de l'installation

A la fin de l'installation, il faut redémarrer l'ordinateur et ensuite on pourra se connecter sur notre compte en tant superviseurci. Nous y avons droit au terminal et nous tapons les commandes suivantes pour installer l'interface graphique :

```
#sudo –i  pour nous connecter en tant que root

#apt-get update

#apt-get install ubuntu-desktop
```

4.6.2 Installation de nagios

Avant d'installer Nagios nous devons disposer :

1. D'un compilateur C (GCC) avec toutes les bibliothèques associées
2. D'une librairie GD (requis pour les scripts CGI)
3. D'un serveur web (apache2)

   Pour cela nous utilisons les commandes suivantes en tant super administrateur (root) :

```
# apt-get install build-essential

# apt-get install gd

# apt-get install apache2
```

A la fin de ces opérations nous créons un compte utilisateur nagios avec un mot de passe

```
#/usr/sbin/useradd –m nagios
```

```
#passwd nagios
```

Création du groupe nagcmd pour autoriser la soumission de commandes externs depuis l'interface web

```
#/usr/sbin/groupadd nagcmd

#/usr/sbin/usermod -G nagcmd nagios

#/usr/sbin/usermod –G nagcmd www-data
```

Téléchargement de nagios et les plugins

```
#mkdir /home/superviseurci/Desktop/téléchargements

#cd /home/superviseurci/Desktop/téléchargements

#wget http://garr.dl.sourceforge.net/project/nagios/nagios-3.x/nagios-3.2.2/nagios-3.2.2.tar.gz

#wget http://garr.dl.sourceforge.net/project/nagiosplug/nagiosplug/1.4.15/nagios-plugins-1.4.15.tar.gz
```

Extraction, compilation et installation de nagios

```
#tar xzvf nagios-3.2.2.tar.gz

#cd nagios-3.2.2

#./configure –with-command-group=nagcmd

#make all

#make install

#make install-init

#make install-config

#make install-commandmode

#vi /usr/local/nagios/etc/objects/contacts.cfg
```

Pour spécifier l'adresse email de l'administrateur pour la notification par e_mail.

Installation du fichier de configuration web Nagios dans le répertoire Apache conf.d puis création d'un compte administrateur nagios pour la connexion à l'interface web

```
#make install-webconf

#htpasswd –c /usr/local/nagios/etc/htpasswd.users nagiosadmin

puis spécifier le mot de passe

#/etc/init.d/apache2 reload
```

Extraction, compilation et installation des plugins

```
#tar xvzf nagios-plugins-1.4.15.tar.gz

#cd nagios-plugins-1.4.15

#./configure –with-nagios-user=nagios –with-nagios-group=nagios

#make

#make install
```

Verification de la configuration et démarrage de Nagios

```
#ln –s /etc/init.d/nagios /etc/rcS.d/S99nagios
```

Pour que nagios s'exécute automatiquement au démarrage du système

```
#/usr/local/nagios/bin/nagios –v /usr/local/nagios/etc/nagios.cfg
```

Pour vérifier les fichiers de configuration de nagios.

Si ok on démarre nagios

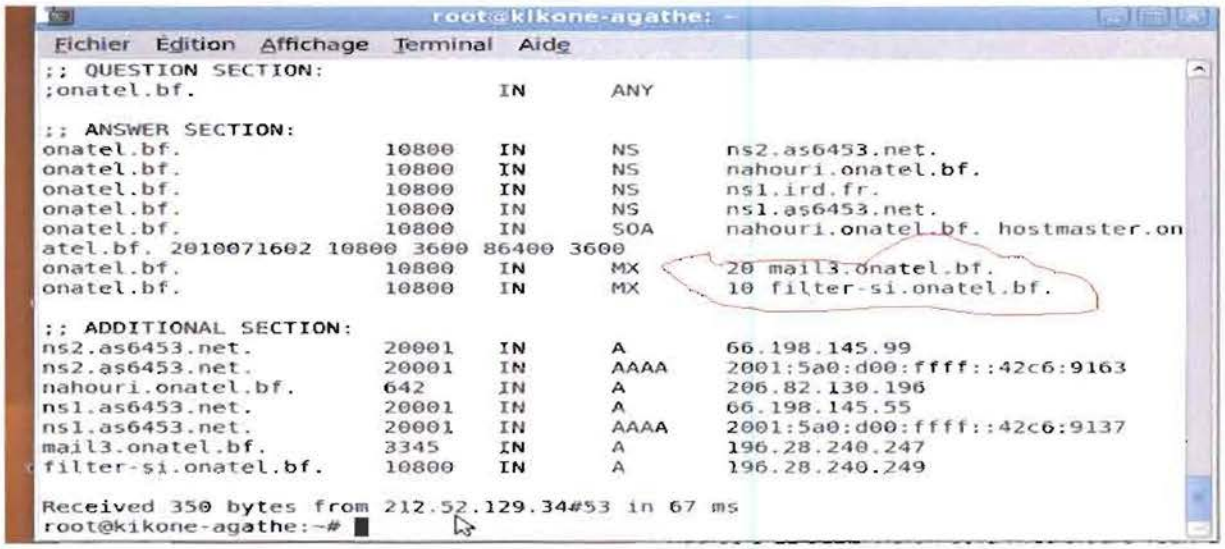
```
#/etc/init.d/nagios start
```

4.6.3 Installation de POSTFIX

Postfix est un système de courrier crée par Wietse Venema, également à l'origine des TCP wrappers, reconnus pour sa large diffusion, ses performances, sa compatibilité avec les autres serveurs de messagerie, sa flexibilité, sa robustesse et sa sécurité. Postfix utilise des requêtes

DNS pour le transfert des mails et nous avons donc besoin du nom de domaine du fournisseur d'accès internet ainsi que son serveur de mail SMTP.

#host –a onatel.bf pour obtenir la liste des hôtes du DNS



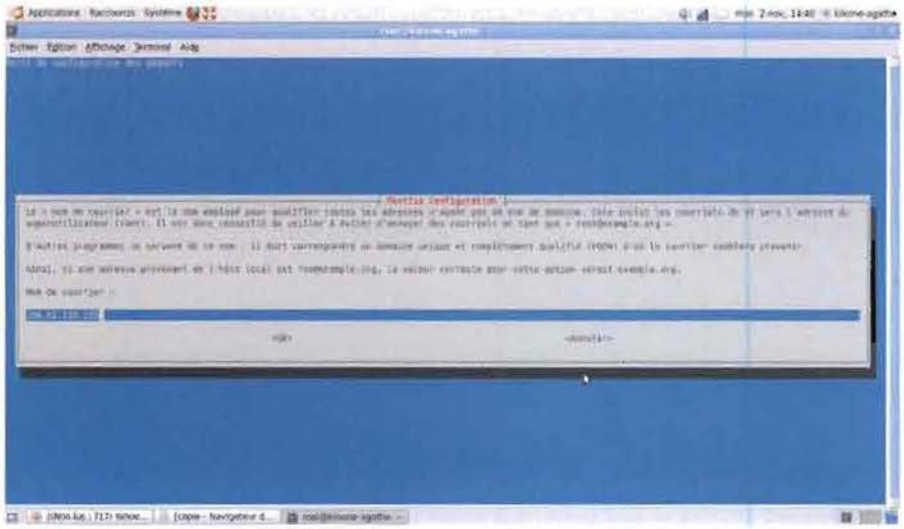
Ici, les serveurs de messagerie dont nous avons besoin sont mail3.onatel.bf et filter-si.onatel.bf

#apt-get install mailx

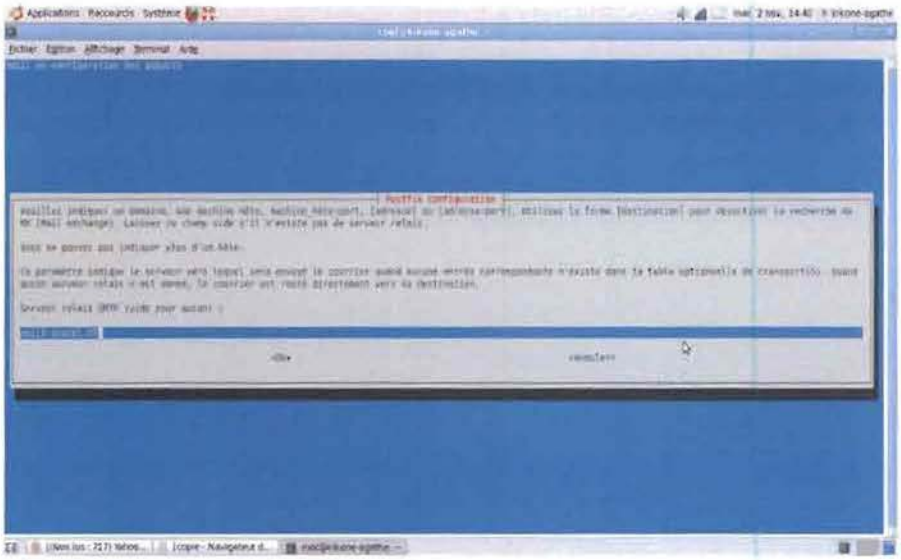
#apt-get install postfix

Pendant l'installation de postfix, nous devons choisir la configuration type du serveur, préciser soit le nom de domaine onatel.bf ou l'adresse IP 206.82.130.195 du serveur DNS et aussi le nom du serveur de messagerie mail3.onatel.bf.

Pour le type de serveur, nous choisissons système satellite car nous avons besoin d'envoyer nos mails via le serveur de messagerie de l'onatel.



Ici, nous précisions simplement l'adresse IP du serveur DNS de l'onatel qui 206.82.130.195



Le serveur relais SMTP que nous allons utiliser est celui de l'onatel qui est mail3.onatel.bf

4.6.4 Installation de NDOUTILS

Avant d'installer NDOUTILS nous devons d'abord installer le serveur de base de données mysql-server, puis les paquets suivant php5, php5-mysql, libmysqlclient15-dev. Nous procédons de la manière suivante :

```
#apt-get install mysql-server

#apt-get install php5

#apt-get install php5-mysql
```

```
#apt-get install libmysqlclient15-dev
```

Ceci étant terminé, nous pouvons télécharger NDOUTILS pour l'installer :

```
#wget http://dfn.dl.sourceforge.net/sourceforge/nagios/ndoutils-1.4b9.tar.gz

#tar zxvf ndoutils-1.4b9.tar.gz

#cd ndoutils-1.4b9

#./configure –prefix=/usr/local/nagios –with-ndo2db-user=nagios –with-ndo2db-group=nagios –enable-mysql

#make all
```

Création de la base de données

```
#mysqladmin –u root –p create centstatus

#mysql –u root –p

#mysql> GRANT CREATE,SELECT,INSERT,DELETE ON centstatus.*TO centreon@localhost IDENTIFY BY 'msconfig';

#mysql>FLUSH PRIVILEGES;

#mysql> EXIT
```

Nous allons maintenant installer le script qui créera automatique les tables dans la base centstatus

```
#cd db

#./installdb –u centreon –p msconfig –h localhost –d centstatus
```

Après cette étape, nous installons les binaires de NDOUTILS présents dans le répertoire src

```
#cd ..

#cd src
```

```
#cp ndomod-3x.o /usr/local/nagios/bin/ndomod-3x.o

#cp ndo2db-3x /usr/local/nagios/bin/ndo2db-3x

#chown nagios:nagios /usr/local/nagios/bin/ndo*
```

Maintenant, nous nous rendons dans le répertoire config pour installer les fichiers de configuration de ndomod et de ndo2db.

```
#cd ..

#cd config

#cp ndomod.cfg-sample /usr/local/nagios/etc/ndomod.cfg

#cp ndo2db.cfg-sample /usr/local/nagios/etc/ndo2db.cfg

#chown nagios:nagios /usr/local/nagios/etc/*.cfg
```

Nous éditons le fichier de configuration ndomod.cfg pour préciser le type de connexion utilisé entre nagios et ndo.

```
#vim /usr/local/nagios/etc/ndomod.cfg

#output_type=tcpsocket

#output=127.0.0.1
```

Ensuite, nous éditons le fichier de configuration ndo2db.cfg pour paramétrer les lignes suivantes :

```
#vim /usr/local/nagios/etc/ndo2db.cfg

#ndo2db_user=nagios

#ndo2db_group=nagios

#socket_type=tcp

#db_user=centreon

#db_pass=msconfig
```

```
#db_name=centstatus

#db_prefix=nagios_
```

Enfin, nous allons éditer le fichier de configuration principal de nagios pour active **l'event_broker**.

```
#vim /usr/local/nagios/etc/nagios.cfg

#event_broker_options=-1

#broker_module=/usr/local/nagios/bin/ndomod.o
config_file=/usr/local/nagios/etc/ndomod.cfg
```

Nous avons finis l'installation et configuration de NDOUTILS et nous allons maintenant le lancer et faire un test pour voir s'il fonctionne bien.

```
#/usr/local/nagios/bin/ndo2db-3x –c /usr/local/nagios/etc/ndo2db.cfg

#/etc/init.d/nagios restart

#tail –f /usr/local/nagios/var/nagios.log
```

resultat obtenu:

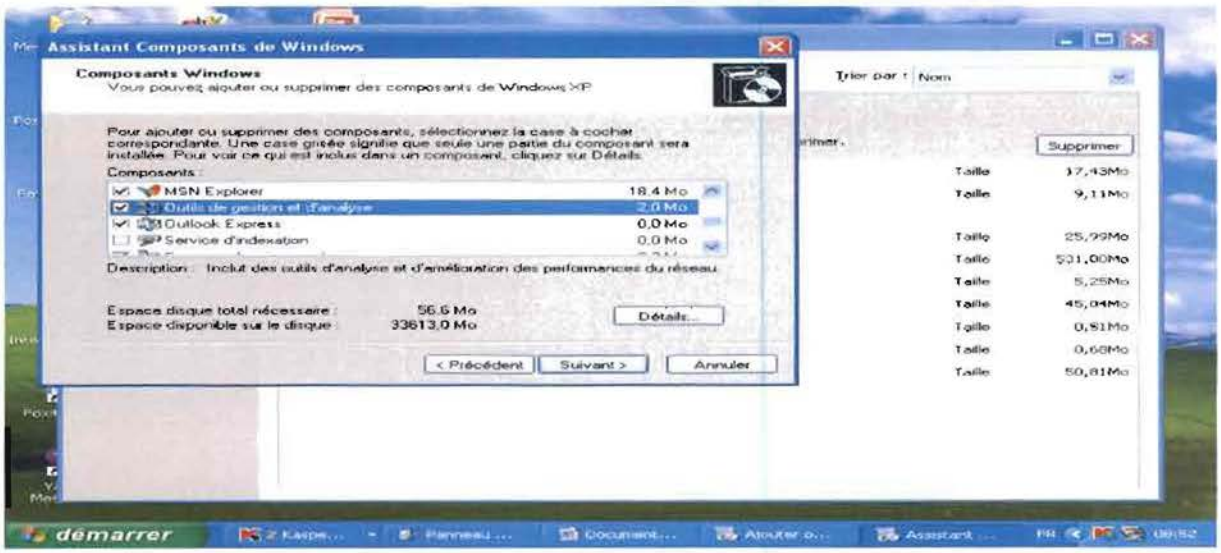
```
[1286967204] Event broker module '/usr/local/nagios/bin/ndomod-3x.o' initialized successfully.
```

4.6.5 Installation de NSClient++ et activation du protocole SNMP

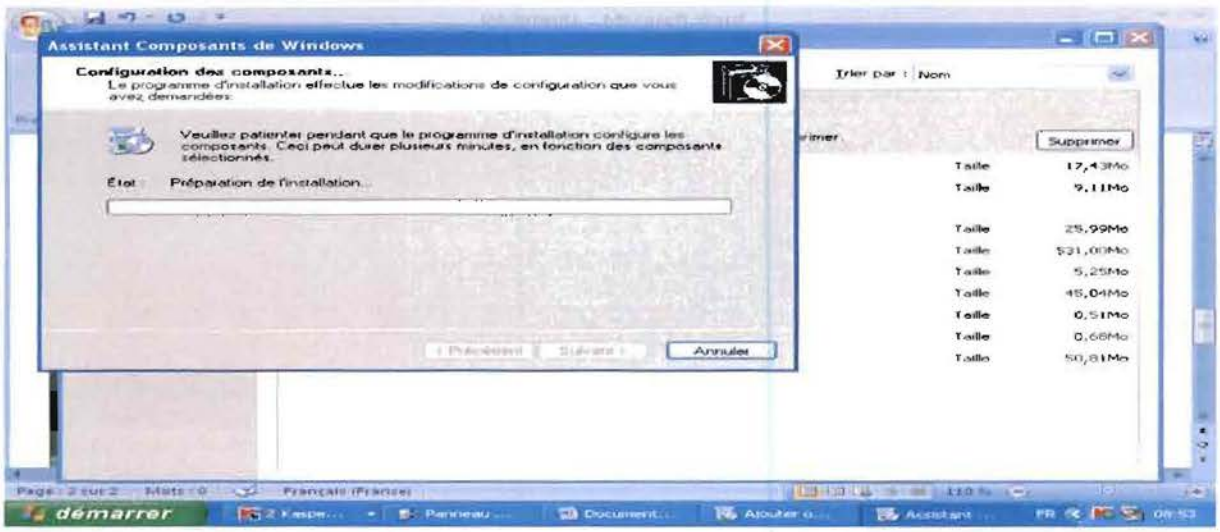
Activation de SNMP

Pour activer le protocole SNMP, nous allons dans **ajout et suppression de programme**, puis nous cliquons sur **ajouter ou supprimer des composants Windows**, à la suite **l'assistant composants de Windows** apparaîtra et nous sélectionnons **outils de gestion et d'analyse** puis sur **suivant** pour que le processus d'installation commence et on attend la fin pour cliquer sur **terminer**. Avant de commencer l'installation nous vérifions d'abord la version du service pack de Windows XP installée pour savoir s'il faut utiliser le CD d'installation de xp ou pas. Si nous sommes face à un service pack 2 de Windows, nous devons insérer obligatoirement le CD d'installation, mais pour les versions de service pack supérieure nous n'en avons pas besoin. Les images suivant relatent l'installation du protocole SNMP.

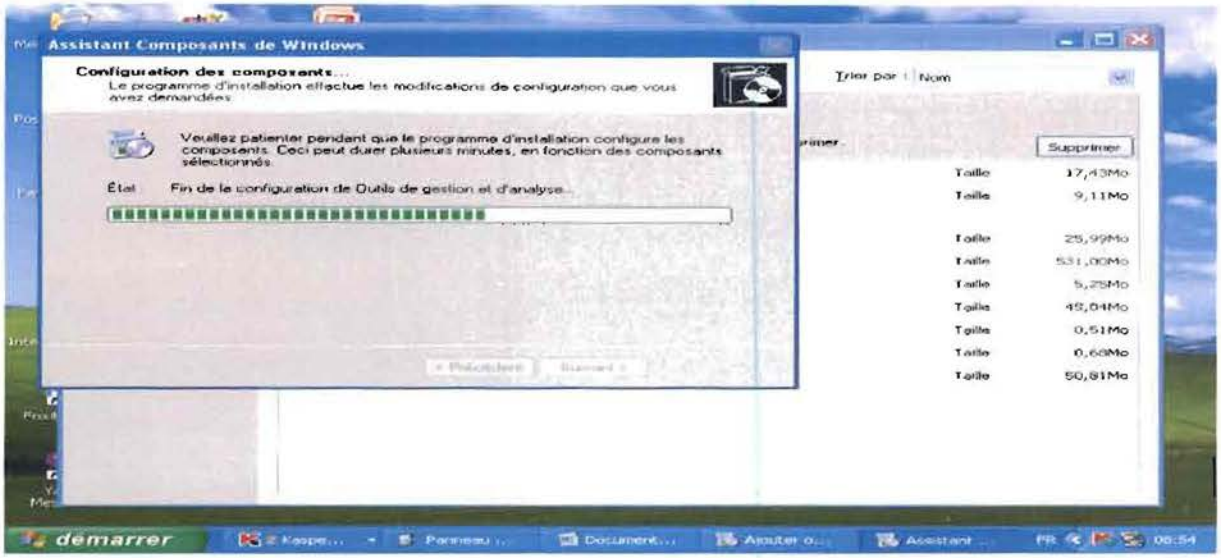
A cette étape, nous cliquons sur **Ajouter ou supprimer des composants Windows**

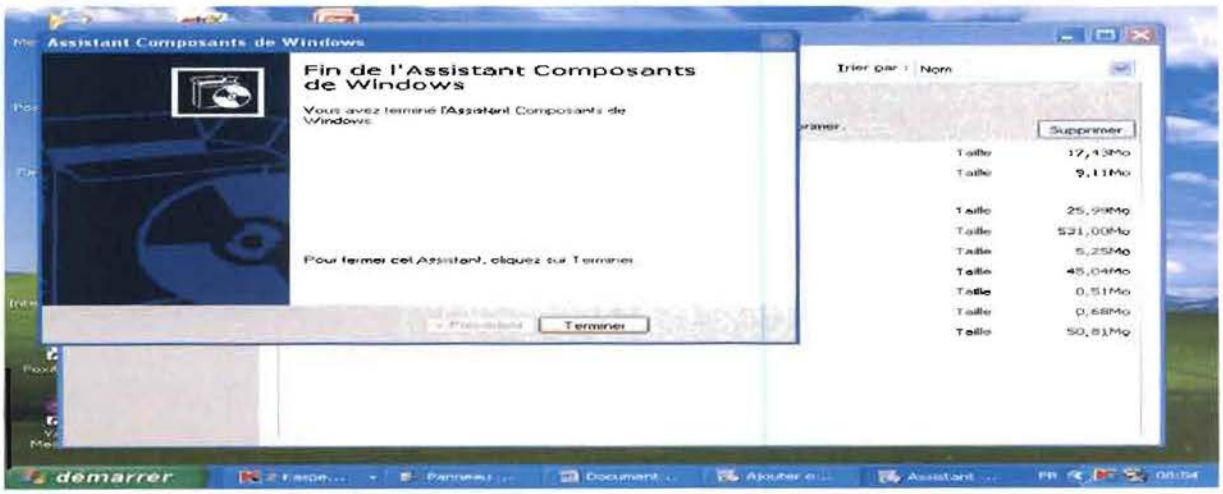


Après avoir sélectionné sur **Outils de gestion et d'analyse**, nous cliquons sur **suivant.**



Le processus d'installation commence.

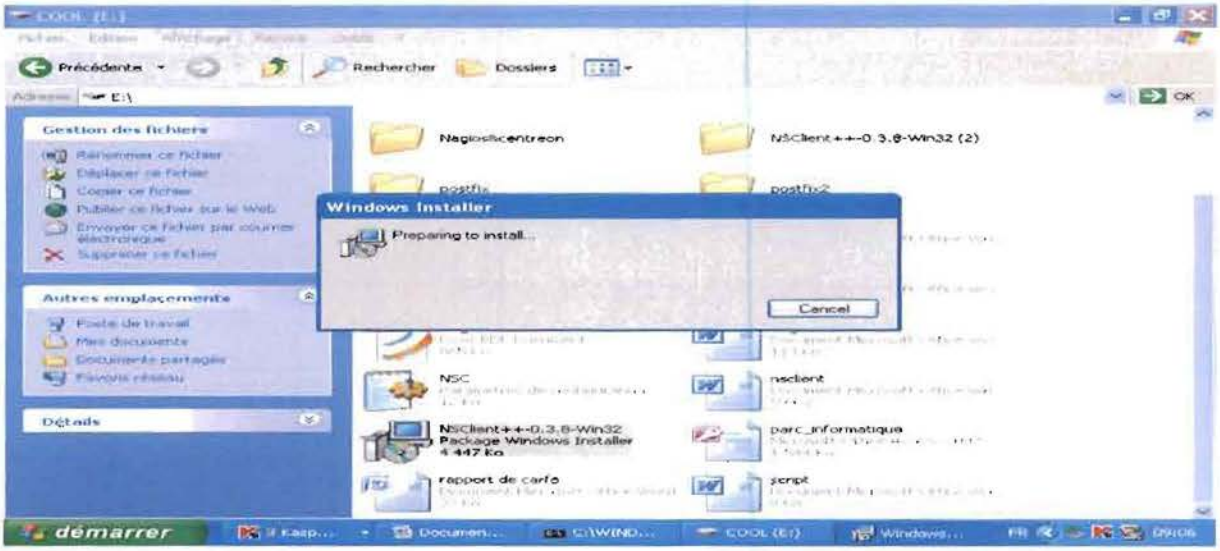




Nous cliquons sur **terminer** pour finir le processus d'installation.

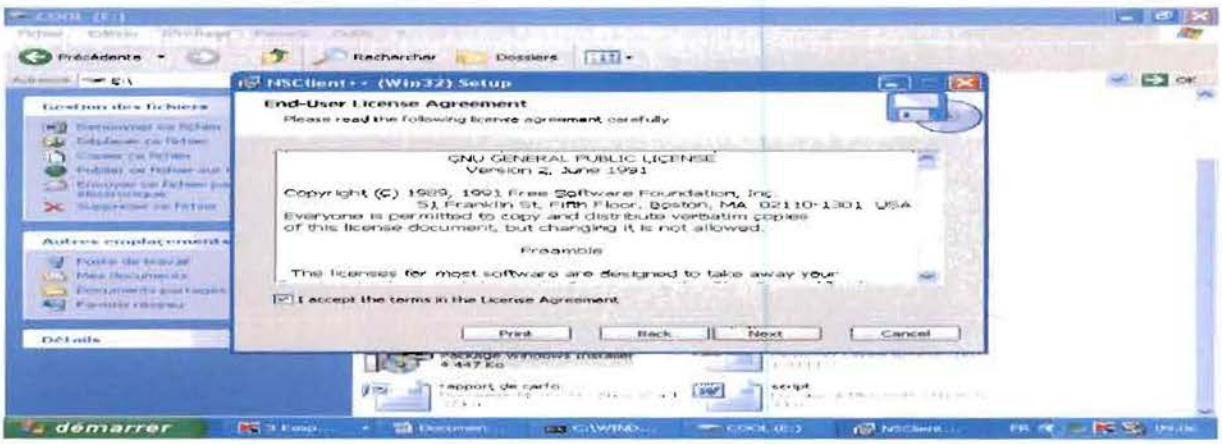
Installation de NSClient++

Pour installer NSClient++, nous avons téléchargé la NSClient++-0.3.8-Win32.msi qui nous offre l'avantage de ne pas désactiver le parfeu Windows après l'installation pour le permettre de fonctionner. Nous résumons l'installation à travers les images suivantes :

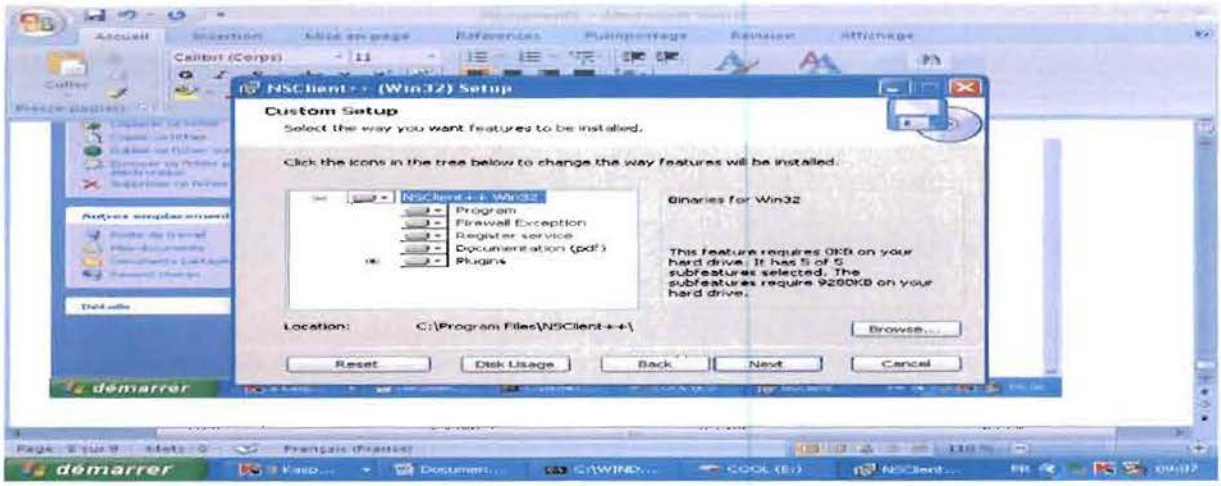


Nous double-cliquons sur NSClient++-0.3.8-Win32.msi pour commencer l'installation.

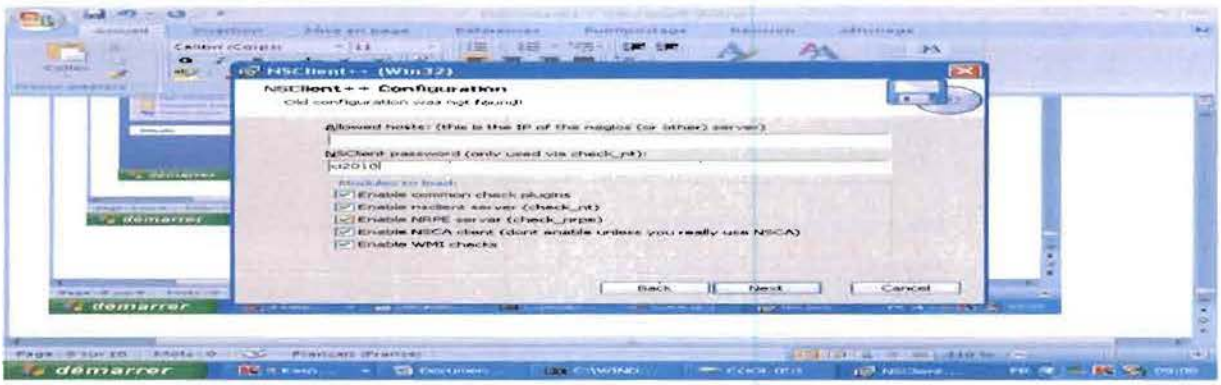
Puis sur **Next** pour continuer l'installation.



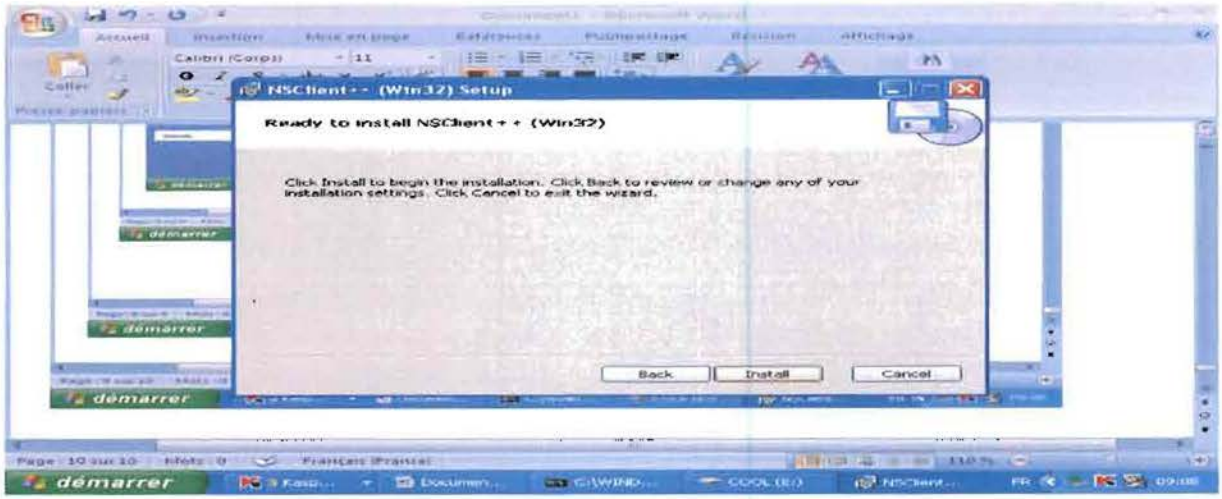
Puis nous acception la licence et nous cliquons sur **Next**



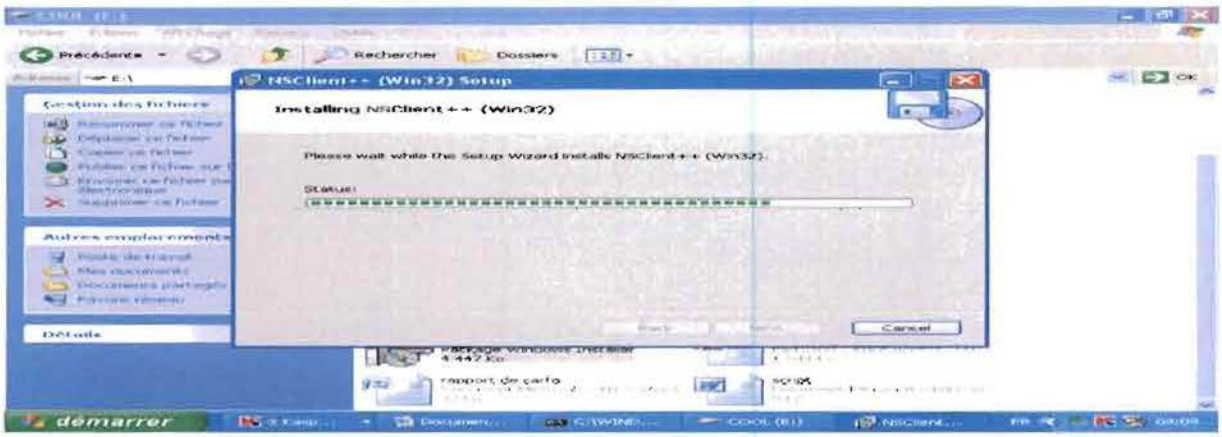
Cette partie nous permet de choisir le répertoire dans lequel nous désirons installer ou accepter le répertoire par défaut qui est C:\Program File\NSClient++. Nous optons pour le répertoire par défaut et nous continuons en cliquant sur **Next.**



Cette fenêtre nous permet de configurer NSClient++ en précisant l'adresse IP du serveur nagios, mais dans notre cas nous laissons la partie vide car nous disposons de 2 serveurs nagios, le mot de passe, et nous activons toutes options de NSClient++. Ceci étant fait nous cliquons sur **Next** pour poursuivre l'installation.



Nous pouvons installer maintenant en cliquant sur **Install.**



L'installation a commencé.

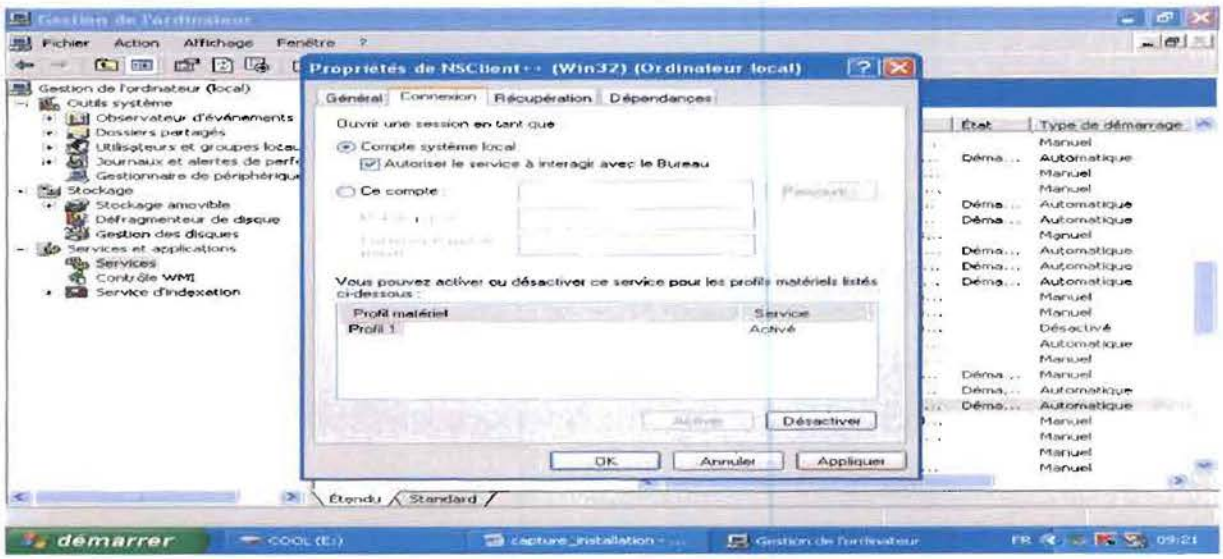
Nous terminons l'installation en cliquant sur Finish, mais avant nous avons cliqué sur Start service pour démarrer le service immédiatement.

Après l'installation de NSClient++, nous nous rendons que le répertoire C:\Program File\NSClient++ pour l'installation de nstray en double-cliquant dessus puis nous éditons le

fichier NSC.ini pour décommenter tous les modules de la section [modules] exceptés CheckWMI.dll et RemoteConfiguration.dll et vérifier que l'option port de la section [NSClient] est décommentée et a pour valeur 12489.

Pour enfin finir avec toutes les manipulations de NSClient++, nous ouvons le gestionnaire des services pour l'autoriser à interagir avec le bureau. Nous procedons comme ceci :

Nous double-cliquons sur le service NSClient++(Win32)



Puis sur **Autoriser le service à interagir avec le Bureau** et **Appliquer.**

Conclusion

Nous pouvons dire que la misse en place d'une solution de supervision nécessite des réflexions et une analyse poussée afin d'adopter une solution qui prenne en compte les besoins de supervision de l'entreprise et qui sera évolutif en fonction de la croissance du

réseau. Ainsi, pour l'élaboration de notre solution de supervision, nous avons fait un inventaire des équipements, réalisée un regroupement des objets en entité logique et concevoir un modèle de fichier pour la supervision des PCs de bureau et des serveurs. Ces étapes ont été bénéfiques pour nous, dans l'élaboration et la mise en œuvre de la solution de supervision.

Vu la mise en œuvre de cette solution, nous pouvons dire qu'elle est fastidieuse car elle demande beaucoup de temps et de recherche. Pour pallier à ce problème, nous proposons d'utiliser FAN (Fully Automated Nagios) qui est une distribution GNU/LINUX et basé sur le système d'exploitation CentOS. FAN permet une installation automatisée de Nagios, Ndoutils, Centreon et Nagvis en vingt minutes.

Conclusion générale

Ce stage de fin de cycle effectué à la CARFO nous a été d'un intérêt capital pour notre formation. Il s'est déroulé en deux phases : une première phase, qui nous a permis de mieux connaître la CARFO, son réseau informatique et de faire des recherches sur notre thème de stage. La deuxième phase, que nous avons appelé phase de test, consistait à mettre en place un serveur nagios pour la supervision de quelques postes afin d'expérimenter ses possibilités et proposer une solution de supervision du réseau.

L'idée du thème de stage était de faire une étude de nagios pour une meilleure exploitation et de proposer une solution de supervision avec ce logiciel afin d'améliorer la gestion du réseau. L'étude de nagios nous à permis de connaître ses possibilités qui sont largement au dessus de nos attentes et de nous rendre compte aussi que nous pouvons toujours l'améliorer en l'associant à une base de données pour avoir un accès rapide au données de supervisons et avec Centreon pour faciliter l'administration. Nous avons donc proposé une solution de supervision haute disponibilité avec nagios qui associe en même temps une base de données MySQL gérer par Ndoutils, et Centreon pour faciliter la configuration des hôtes. La mise en œuvre de cette solution est un peu complexe car nous avons deux serveurs nagios pour assurer la supervision qui fonctionnent en maître/esclave mais permet une supervision de réseau à tout moment.

La CARFO n'est pas en marge du phénomène de la décentralisation qui est en cours dans notre pays et qui concerne tous les secteurs d'activés. C'est ainsi que la CARFO possède des structures déconcentrées qui sont les directions régionales qui ont aussi besoin des services de la cellule informatique pour fonctionner. Étant donné que la supervision doit évoluer en même temps que le réseau, une solution de supervision distribuée peut être intégrer pour prendre en compte les réseaux des directions régionales.

Ce projet nous a permis de consolider nos connaissances en administration linux et réseaux, et de faire une prospection dans le monde de la supervision. Nous suggérons néanmoins un renforcement de la politique de sécurité des données avec une migration des serveurs qui sont sous Windows 2003 server vers des servers linux qui offrent une garantie solide en matière de sécurité et qui ont aussi pour avantage d'être compatible avec les autres systèmes d'exploitation.

Bibliographie et webographie

Bibliographie

Note de cours d'administration réseau du COLONEL AOUBA MOHAMADI

Mémoire DESS [2005-2006] de Monsieur BADO Noël au format PDF

Mémoire CICI [2000-2001] de Monsieur DAVOU Moussa au format PDF

Rapport de stage CITI [2008-2009] au format PDF de Mr SANON Dramane Edmond

Rapport de stage CITI [2008-2009] au format PDF de Mr DRABO L Wilfried

Rapport de stage au format PDF de Mr BOUSSALEM Assam et de ZERARA Younes

Livre : Quels sont les outils d'aujourd'hui de supervision réseau LAN/WAN tome IV de Jean-François CASQUET au format PDF

Livre : Nagios 3 pour la supervision et la métrologie au format PDF de Jean GABES

Documentation Nagios version 3x en français

Webographie

http://christian.caleca.free.fr/tcpip consulté le 25août 2010

http://www.commentcamarche.com consulté du 12 août au 25 novembre 2010

http://blog.nicolargo.com consulté du 12 août au 25 novembre 2010

http://www.monitoring-fr.org consulté du 12 août au 25 novembre 2010

http://www.ubuntu-fr.com consulté du 12 août au 25 novembre 2010

http://exchange.nagios.org consulté du 12 août au 25 novembre 2010

http://www.centreon.com consulté du 12 août au 25 novembre 2010

http://packages.debian.org consulté le 22 septembre 2010

http://www.wikipedia.fr consulté du 12 août au 25 novembre 2010

http://www.carfo.fr consulté du 22 au 24 novembre 2010

Table des matières

DEDICACE i

Remerciements 2

Avant propos 3

Sommaire 4

Sigles et abréviations 6

Introduction Générale 7

Chapitre 1 : Présentation de la CARFO et de son réseau informatique. 8

Introduction 8

1.1Historique 8

1.2Missions et attributions 9

1.3Orientations stratégiques 10

1.4 Organigramme de la CARFO 10

1.5 Le réseau informatique 11

Conclusion 14

Chapitre 2 : La supervision informatique 15

Introduction 15

2.1 Objectifs et avantages 15

2.2 Méthodes de supervision informatique 15

2.3 Le protocole SNMP 16

2.3.1 Historique 16

2.3.2 Fonctionnement 17

2.4 Les moniteurs de supervisions 20

2.4.1 Les moniteurs de supervision payant 20

2.4.2 Les moniteurs de supervision libre 21

Conclusion 22

Chapitre 3 : Etude de Nagios 23
Introduction 23
3.1 Historique 23
3.2 Fonctionnalités 23
3.3 Architecture 24
3.4 Les principes de base de Nagios 29
3.4.1 Détermination de l'état et du type d'état des hôtes 29
3.4.1.1 Détermination de l'état des hôtes et/ou des services du réseau 29
3.4.1.2 Détermination du type d'état des hôtes et/ou des services du réseau 30
3.4.2 Détermination de l'accessibilité des hôtes du réseau 31
3.4.3 Le contrôle des hôtes et des services 32
3.4.3.1 Les contrôles actifs 32
3.4.3.2 Les contrôles passifs 33
3.4.4 Les notifications 33
3.4.5 Les périodes de temps 35
3.6 Installation 35
3.7 Configuration 36
3.7.1 Le fichier de configuration principal 37
3.7.2 Le fichier de configuration des ressources 37
3.7.4 Le fichier de configuration des CGIs 38
3.8 La supervision des éléments du réseau 38
3.8.1 Supervision des machines Windows 38
3.8.2 Supervision des machines Linux/Unix 38
3.8.3 Supervision des imprimantes réseau 39
3.8.4 Supervision des routeurs et des switchs 39
3.9 Les types de supervision 40
3.9.1 La supervision distribuée 40

3.9.2 La supervision redondante 41
3.9.3 La supervision haute disponibilité 42
Conclusion 43
Chapitre 4 : proposition de la solution de supervision et mise en œuvre 44
Introduction 44
4.1 Inventaire des équipements du réseau 44
4.2 Détermination des objets à prendre en compte pour la supervision 45
4.3 Regroupement des objets en entités logique 46
4.3.1 Modèles 46
4.3.2 Fichier de configuration des serveurs et des PCs de bureau 46
4.4 Proposition de la solution de supervision 48
4.4.1 La supervision haute disponibilité pour nagios 48
4.4.2 Ndoutils 51
4.4.2 Centreon 52
4.5 Les besoins 53
4.6 Mise en œuvre 53
4.6.1 Installation du système 53
4.6.2 Installation de nagios 54
4.6.3 Installation de POSTFIX 56
4.6.4 Installation de NDOUTILS 58
4.6.5 Installation de NSClient++ et activation du protocole SNMP 61
Activation de SNMP 61
Installation de NSClient++ 63
Conclusion 67
Conclusion générale 69
Bibliographie et webographie 70
Table des matières 71

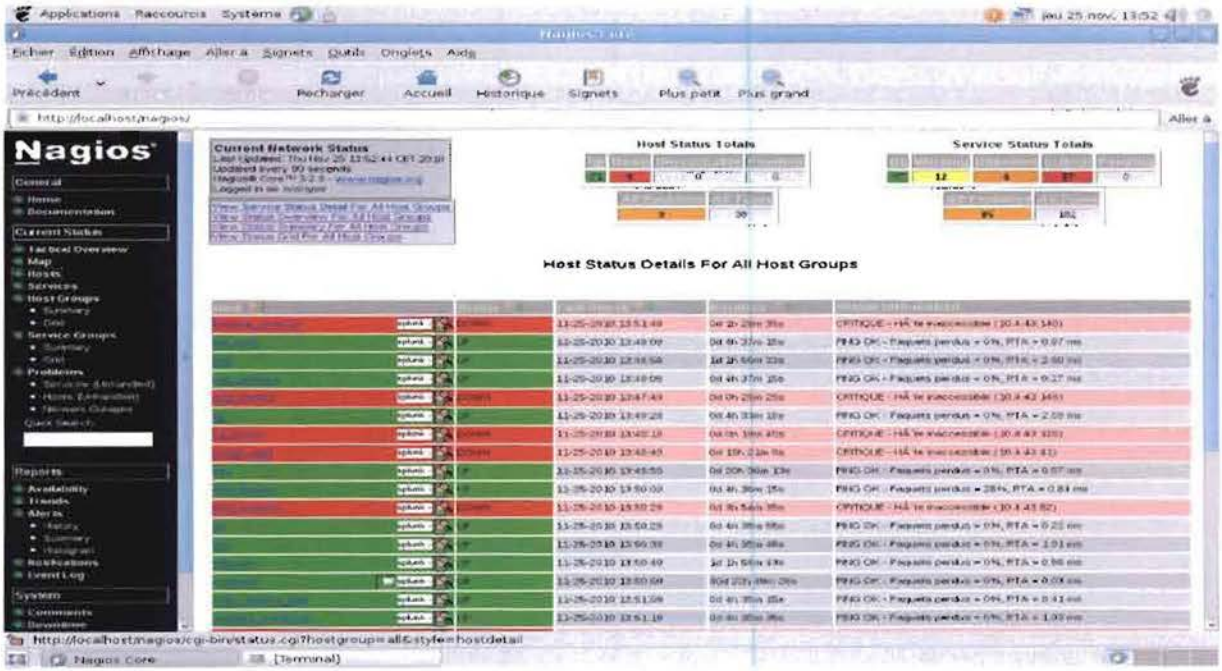
Annexe 75

1. Les interfaces de Nagios 75
2. Organigramme de la CARFO 78
3. Contenu du fichier /usr/local/nagios/etc/objects/contacts.cfg 80
4. Contenu du fichier /usr/local/nagios/etc/objects/commandes.cfg 81
5. Contenu du fichier /usr/local/nagios/etc/objects/localhost.cfg 86
6. Contenu de fichier NSC.INI 90

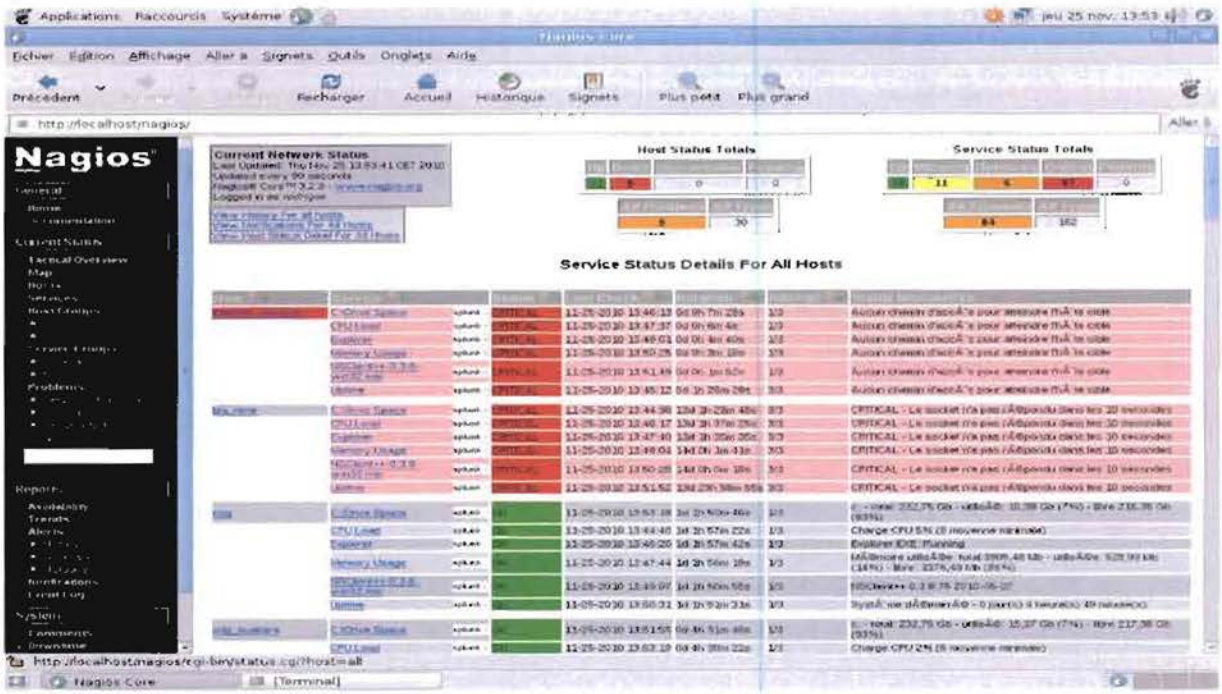
Annexe

1. Les interfaces de Nagios

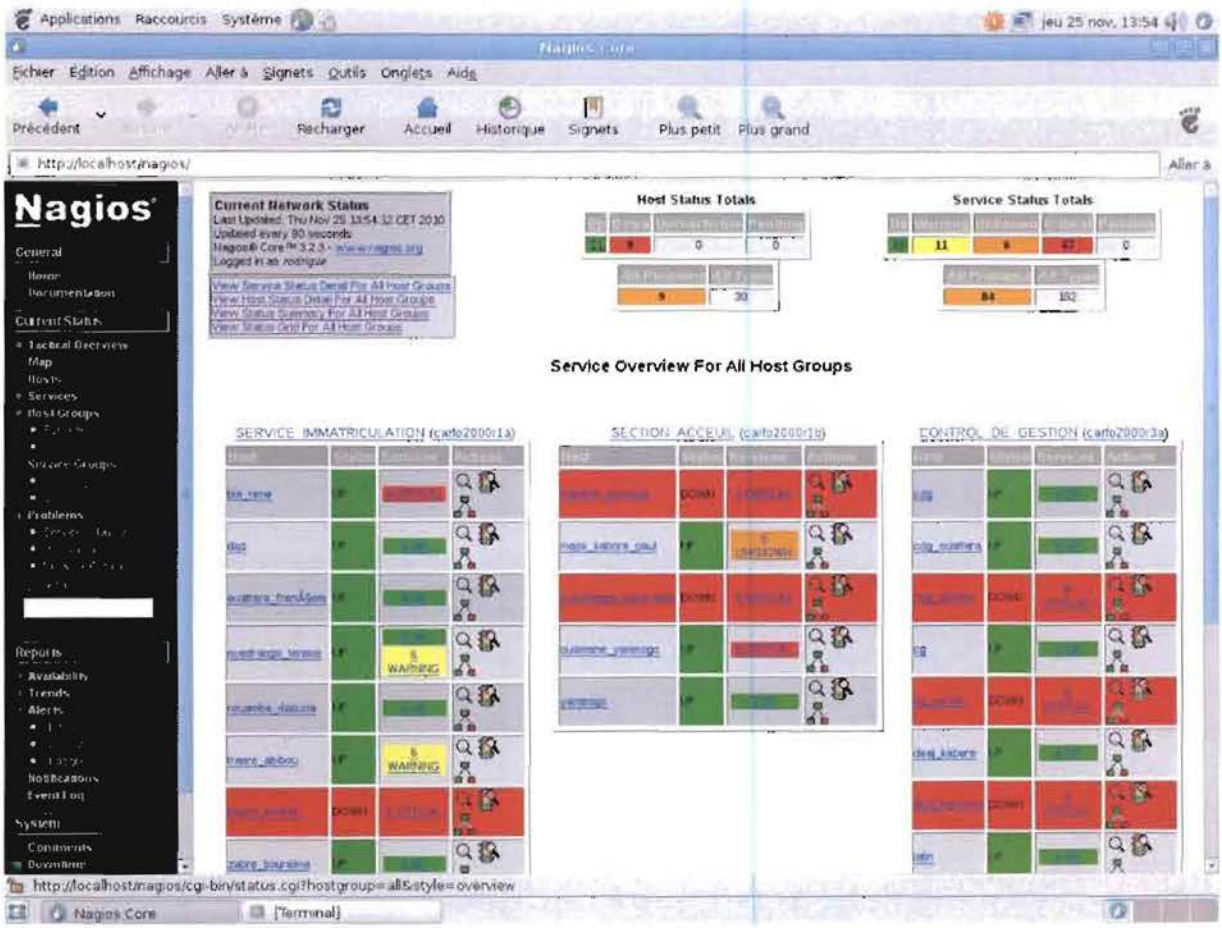
- Vue du statut des hôtes



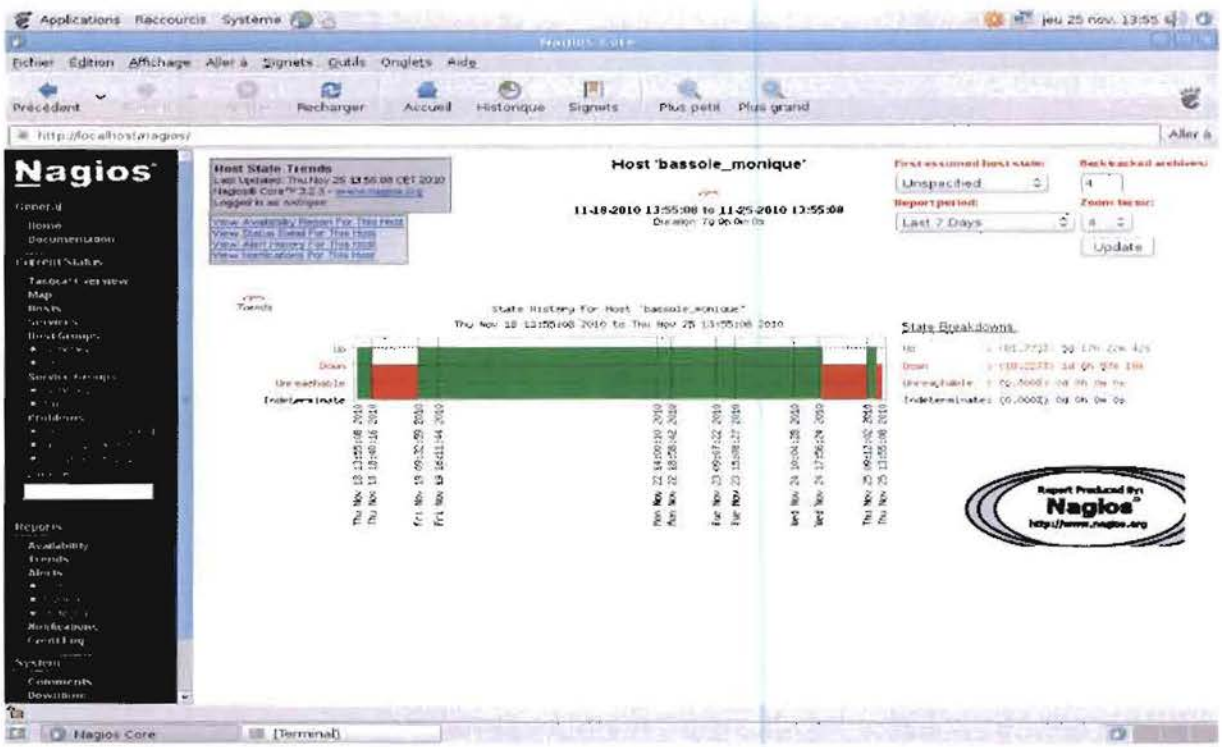
- Vue du statut des services pour chaque hôte



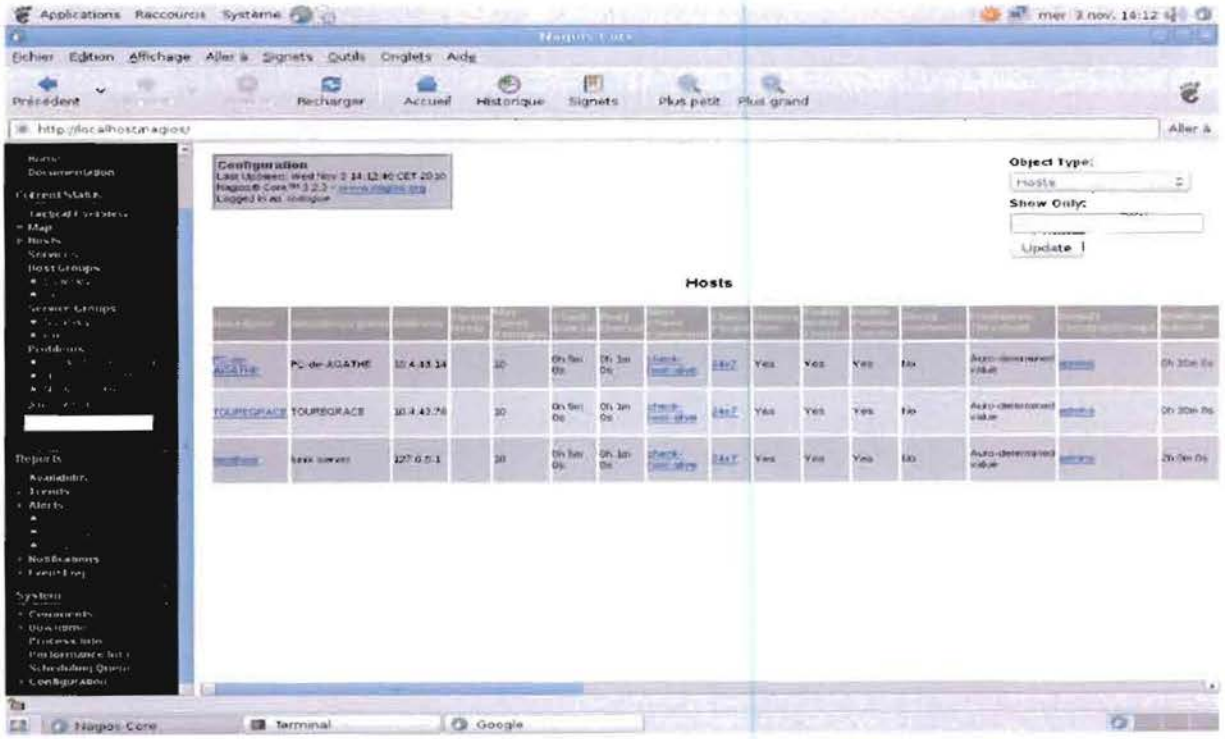
- Vue du statut des groupes d'hôtes



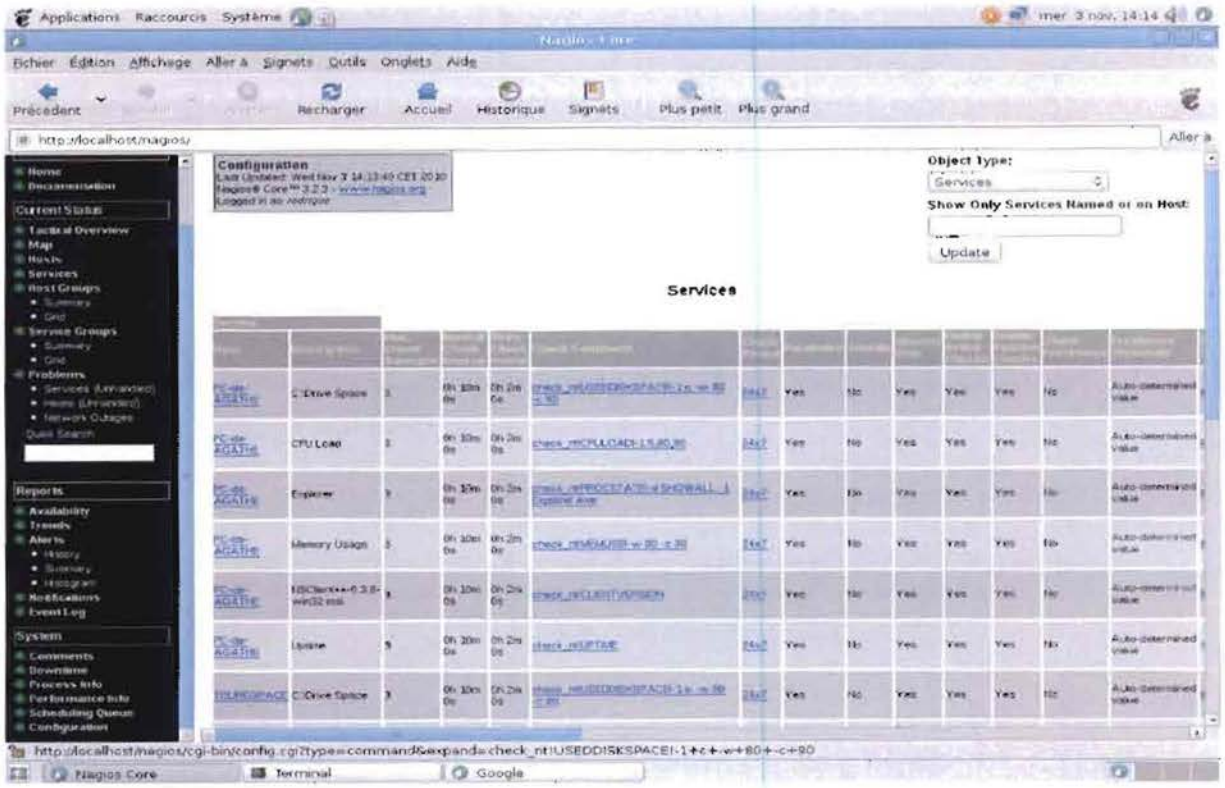
- Vue du repporting



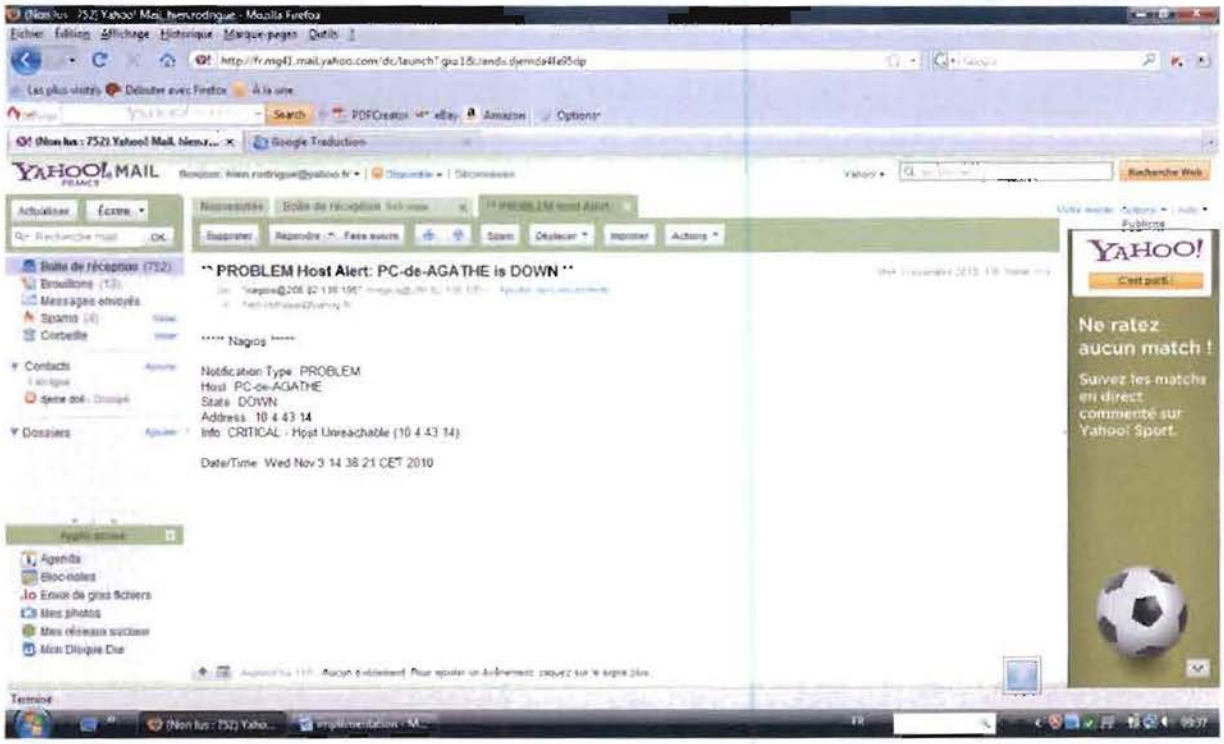
- Vue de la configuration des hôtes

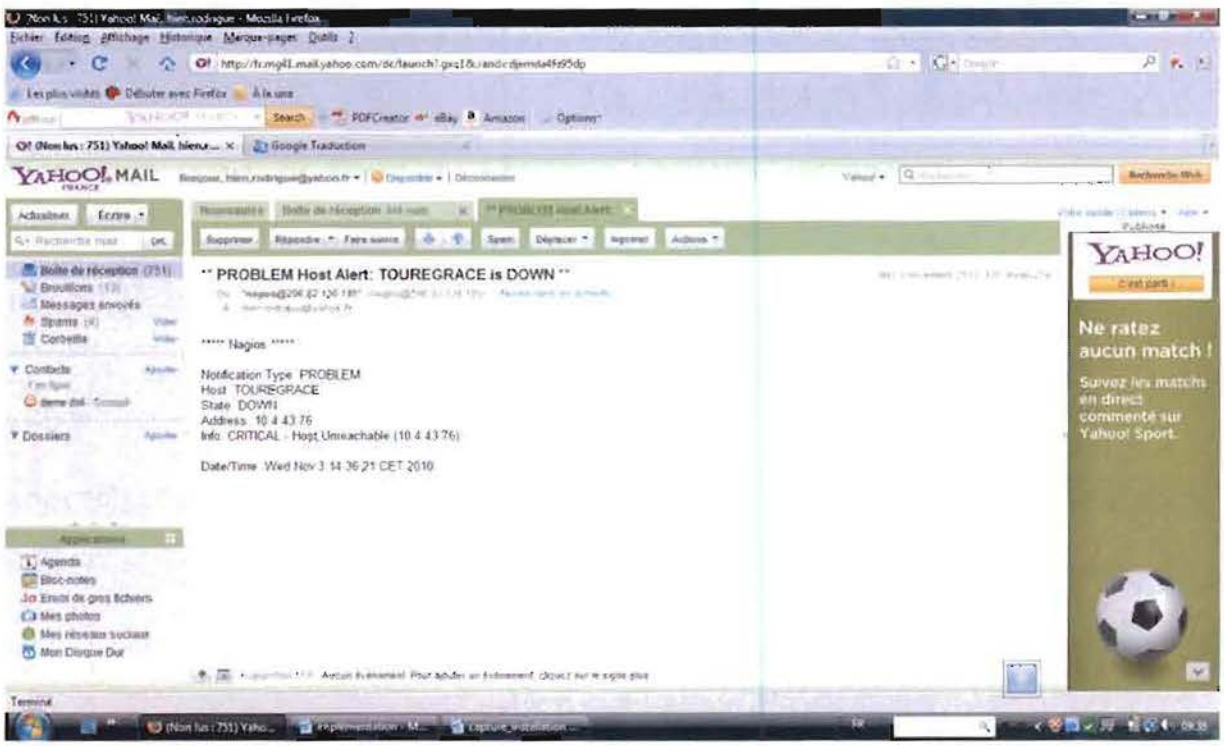


- Vue de configuration des services



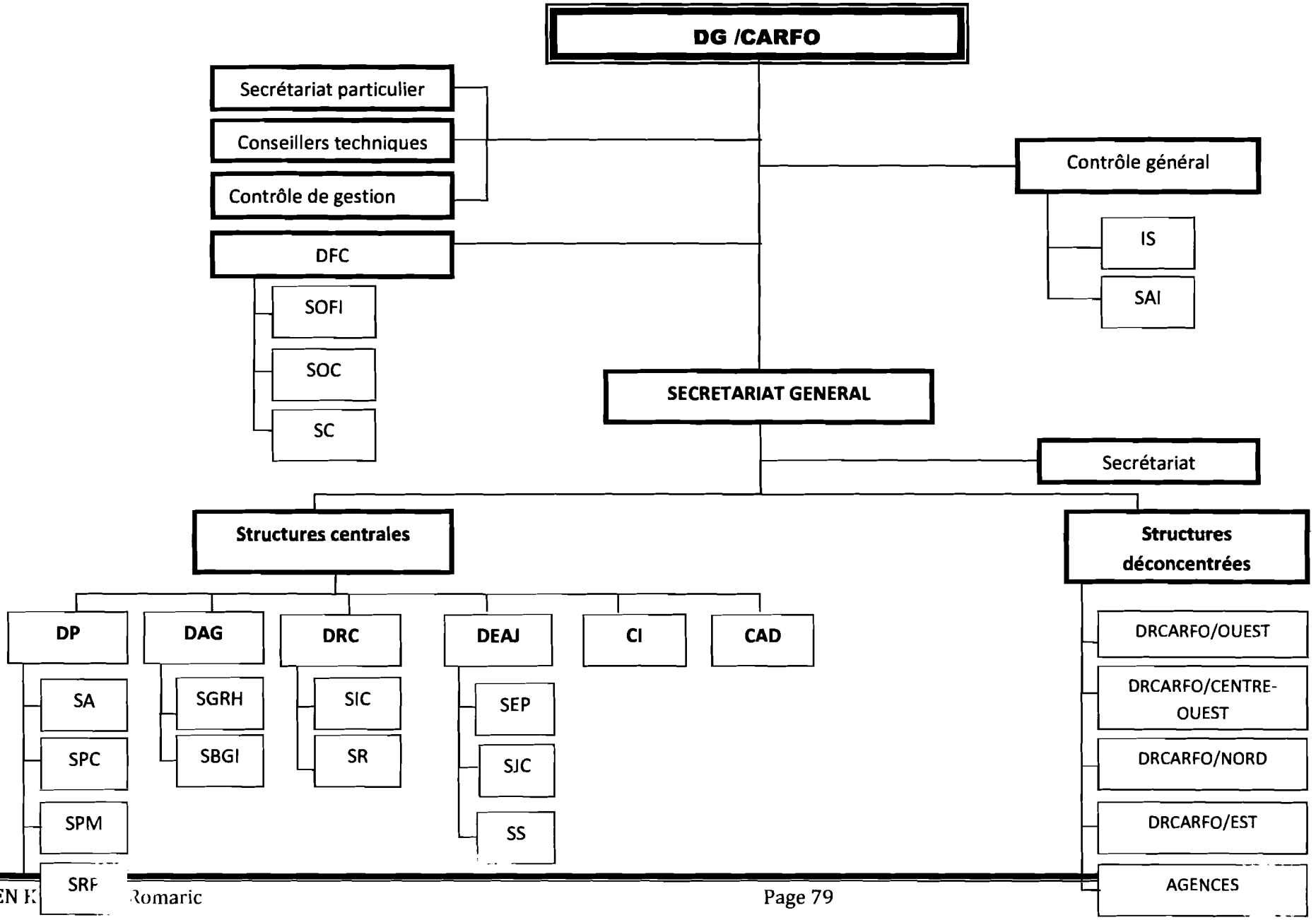
- Notification envoyée par Nagios





- Organigramme de la CARFO

2. Organigramme de la CARFO


DG /CARFO
Secrétariat particulier
Conseillers techniques
Contrôle de gestion
DFC
SOFI
SOC
SC
Contrôle général
IS
SAI
SECRETARIAT GENERAL
Secrétariat
Structures centrales
Structures déconcentrées
DP
SA
SPC
SPM
SRF
DAG
SGRH
SBGI
DRC
SIC
SR
DEAJ
SEP
SJC
SS
CI
CAD
DRCARFO/OUEST
DRCARFO/CENTRE-OUEST
DRCARFO/NORD
DRCARFO/EST
AGENCES

3. Contenu du fichier /usr/local/nagios/etc/objects/contacts.cfg

```
##############################################################################
# CONTACTS.CFG - SAMPLE CONTACT/CONTACTGROUP DEFINITIONS #
# Last Modified: 05-31-2007#
# NOTES: This config file provides you with some example contact and contact
#       group definitions that you can reference in host and service
#       definitions.
#      #       You don't need to keep these definitions in a separate file from your
#       other object definitions.  This has been done just to make things
#       easier to understand.
##############################################################################
##############################################################################
##############################################################################
#
# 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             ; Short name of user
        use                                 generic-contact          ; Inherit default values from generic-
contact template (defined above)
      alias                   Nagios Admin              ; Full name of user
      email                    hien.rodrigue@yahoo.fr     ; <<***** CHANGE THIS TO YOUR
EMAIL ADDRESS ******
      }
##############################################################################
##############################################################################
## 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
    }
```

4. Contenu du fichier /usr/local/nagios/etc/objects/commandes.cfg

```
##############################################################################
# COMMANDS.CFG - SAMPLE COMMAND DEFINITIONS FOR NAGIOS 3.2.3
#
# Last Modified: 05-31-2007
#
# NOTES: This config file provides you with some example command definitions
#       that you can reference in host, service, and contact definitions.
#       You don't need to keep commands in a separate file from your other
#       object definitions.  This has been done just to make things easier to
#       understand.
##############################################################################
##############################################################################
#
# SAMPLE NOTIFICATION COMMANDS
#
# These are some example notification commands.  They may or may not work on
# your system without modification.  As an example, some systems will require
# you to use "/usr/bin/mailx" instead of "/usr/bin/mail" in the commands below.
##############################################################################
# 'notify-host-by-email' command definition
define command{
        command_name            notify-host-by-email
        command_line  /usr/bin/printf "%b" "***** Nagios *****\n\nNotification Type:
$NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAddress:
$HOSTADDRESS$\nInfo: $HOSTOUTPUT$\n\nDate/Time: $LONGDATETIME$\n" | /usr/bin/mail
```

```
-s "** $NOTIFICATIONTYPE$ Host Alert: $HOSTNAME$ is $HOSTSTATE$ **"
$CONTACTEMAIL$
        }
# 'notify-service-by-email' command definition
define command{
        command_name          notify-service-by-email
        command_line /usr/bin/printf "%b" "***** Nagios *****\n\nNotification Type:
$NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost: $HOSTALIAS$\nAddress:
$HOSTADDRESS$\nState: $SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\nAdditional
Info:\n\n$SERVICEOUTPUT$\n" | /usr/bin/mail -s "** $NOTIFICATIONTYPE$ Service Alert:
$HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ **" $CONTACTEMAIL$
        }
################################################################################
# SAMPLE HOST CHECK COMMANDS
#
################################################################################
# This command checks to see if a host is "alive" by pinging it
# The check must result in a 100% packet loss or 5 second (5000ms) round trip


# average time to produce a critical error.
# Note: Five ICMP echo packets are sent (determined by the '-p 5' argument)
# 'check-host-alive' command definition
define command{
    command_name    check-host-alive
    command_line    $USER1$/check_ping -H $HOSTADDRESS$ -w 3000.0,80% -c 5000.0,100% -
p 5
    }
################################################################################
# SAMPLE SERVICE CHECK COMMANDS
#
# These are some example service check commands.  They may or may not work on
# your system, as they must be modified for your plugins.  See the HTML
# documentation on the plugins for examples of how to configure command definitions.#
# NOTE:  The following 'check_local_...' functions are designed to monitor
#       various metrics on the host that Nagios is running on (i.e. this one).
################################################################################
```

```
#'check_local_disk' command definition
define command{
        command_name    check_local_disk
        command_line    $USER1$/check_disk -w $ARG1$ -c $ARG2$ -p $ARG3$
        }
# 'check_local_load' command definition
define command{
        command_name    check_local_load
        command_line    $USER1$/check_load -w $ARG1$ -c $ARG2$
        }
# 'check_local_procs' command definition
define command{
        command_name    check_local_procs
        command_line    $USER1$/check_procs -w $ARG1$ -c $ARG2$ -s $ARG3$
        }
# 'check_local_users' command definition
define command{
        command_name    check_local_users

        command_line    $USER1$/check_users -w $ARG1$ -c $ARG2$
        }
# 'check_local_swap' command definition
define command{
            command_name            check_local_swap
            command_line  $USER1$/check_swap -w $ARG1$ -c $ARG2$
            }
# 'check_local_mrtgtraf' command definition
define command{
            command_name            check_local_mrtgtraf


command_line  $USER1$/check_mrtgtraf -F $ARG1$ -a $ARG2$ -w $ARG3$ -c $ARG4$ -e
$ARG5$
            }
################################################################################
#NOTE:  The following 'check_...' commands are used to monitor services on
#       both local and remote hosts.
################################################################################
```

```
# 'check_ftp' command definition
define command{
        command_name    check_ftp
        command_line    $USER1$/check_ftp -H $HOSTADDRESS$ $ARG1$
        }
# 'check_hpjd' command definition
define command{
        command_name    check_hpjd
        command_line    $USER1$/check_hpjd -H $HOSTADDRESS$ $ARG1$

# 'check_snmp' command definition

    define command{
        command_name    check_snmp
        command_line    $USER1$/check_snmp -H $HOSTADDRESS$ $ARG1$
        }
# 'check_http' command definition
define command{
        command_name    check_http
        command_line    $USER1$/check_http -I $HOSTADDRESS$ $ARG1$
        }
# 'check_ssh' command definition
define command{
        command_name            check_ssh
        command_line  $USER1$/check_ssh $ARG1$ $HOSTADDRESS$
        }
# 'check_dhcp' command definition
define command{
        command_name            check_dhcp
        command_line  $USER1$/check_dhcp $ARG1$
        }
# 'check_ping' command definition
define command{
        command_name    check_ping
        command_line    $USER1$/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$ -p 5
        }
#'check_nrpe' command definition
```

```
define command {
        command_name            check_nrpe
 command_line            $USER1$/check_nrpe -H $HOSTADDRESS$ -c $ARG1$
}
# 'check_pop' command definition
define command{
    command_name    check_pop
    command_line    $USER1$/check_pop -H $HOSTADDRESS$ $ARG1$
    }
# 'check_imap' command definition
define command{
    command_name    check_imap
    command_line    $USER1$/check_imap -H $HOSTADDRESS$ $ARG1$
    }
# 'check_smtp' command definition
define command{
    command_name    check_smtp
    command_line    $USER1$/check_smtp -H $HOSTADDRESS$ $ARG1$
    }
# 'check_tcp' command definition
define command{
        command_name        check_tcp
        command_line  $USER1$/check_tcp -H $HOSTADDRESS$ -p $ARG1$ $ARG2$
        }
# 'check_udp' command definition
define command{
        command_name        check_udp
        command_line  $USER1$/check_udp -H $HOSTADDRESS$ -p $ARG1$ $ARG2$
        }
# 'check_nt' command definition
define command{
        command_name        check_nt

        command_line  $USER1$/check_nt -H $HOSTADDRESS$ -p 12489 -s dxdiag -v $ARG1$
$ARG2$
        }
##############################################################################
```

```
# SAMPLE PERFORMANCE DATA COMMANDS
#
# These are sample performance data commands that can be used to send performance
# data output to two text files (one for hosts, another for services). If you
# plan on simply writing performance data out to a file, consider using the
# host_perfdata_file and service_perfdata_file options in the main config file.
##############################################################################
# 'process-host-perfdata' command definition
define command{
        command_name            process-host-perfdata

command_line /usr/bin/printf "%b"
"$LASTHOSTCHECK$\t$HOSTNAME$\t$HOSTSTATE$\t$HOSTATTEMPT$\t$HOSTSTATETY
PE$\t$HOSTEXECUTIONTIME$\t$HOSTOUTPUT$\t$HOSTPERFDATA$\n" >>
/usr/local/nagios/var/host-perfdata.out
        }
# 'process-service-perfdata' command definition
define command{
        command_name            process-service-perfdata
        command_line /usr/bin/printf "%b"
"$LASTSERVICECHECK$\t$HOSTNAME$\t$SERVICEDESC$\t$SERVICESTATE$\t$SERVICE
ATTEMPT$\t$SERVICESTATETYPE$\t$SERVICEEXECUTIONTIME$\t$SERVICELATENCY$\
t$SERVICEOUTPUT$\t$SERVICEPERFDATA$\n" >> /usr/local/nagios/var/service-perfdata.out
        }
```

5. Contenu du fichier /usr/local/nagios/etc/objects/localhost.cfg

```
##############################################################################
# LOCALHOST.CFG - SAMPLE OBJECT CONFIG FILE FOR MONITORING THIS MACHINE
#
# Last Modified: 05-31-2007
#
# NOTE: This config file is intended to serve as an *extremely* simple
#       example of how you can create configuration entries to monitor
#       the local (Linux) machine.
```

```
##############################################################################
##############################################################################
##############################################################################
# HOST DEFINITION
#
##############################################################################
##############################################################################

# Define a host for the local machine
define host{
    use             linux-server        ; Name of host template to use
                                        ; This host definition will inherit all variables that are
defined                                 ; in (or inherited by) the linux-server host template
definition.
host_name           localhost
    alias           lunix server
    address         127.0.0.1
      icon_image            server-nagios.png
      statusmap_image               server-nagios.gd2

 }
##############################################################################
##############################################################################
# HOST GROUP DEFINITION
##############################################################################
##############################################################################
# Define an optional hostgroup for Linux machines
define hostgroup{
    hostgroup_name  linux-servers ; The name of the hostgroup
    alias       Linux Servers ; Long name of the group
    members     localhost   ; Comma separated list of hosts that belong to this group
    }
##############################################################################
##############################################################################
# SERVICE DEFINITIONS
#
##############################################################################
##############################################################################
```

```
# Define a service to "ping" the local machine
define service{
    use                     local-service       ; Name of service template to use
    host_name               localhost
    service_description         PING
      check_command                         check_ping!100.0,20%!500.0,60%
    }
# Define a service to check the disk space of the root partition
# on the local machine.  Warning if < 20% free, critical if
# < 10% free space on partition.
define service{
    use                     local-service       ; Name of service template to use
    host_name               localhost
    service_description         Root Partition
      check_command                         check_local_disk!20%!10%!/
    }
# Define a service to check the number of currently logged in

# users on the local machine.  Warning if > 20 users, critical
# if > 50 users.
define service{
    use                     local-service       ; Name of service template to use
    host_name               localhost
    service_description         Current Users
      check_command                         check_local_users!20!50
    }
# Define a service to check the number of currently running procs
# on the local machine.  Warning if > 250 processes, critical if

# > 400 users.
define service{
    use                     local-service       ; Name of service template to use
    host_name               localhost
    service_description         Total Processes
      check_command                         check_local_procs!250!400!RSZDT
    }
# Define a service to check the load on the local machine.
define service{
```

```
    use                     local-service       ; Name of service template to use
    host_name               localhost
    service_description     Current Load
      check_command                     check_local_load!5.0,4.0,3.0!10.0,6.0,4.0
    }
# Define a service to check the swap usage the local machine.
# Critical if less than 10% of swap is free, warning if less than 20% is free
define service{
    use                     local-service       ; Name of service template to use
    host_name               localhost
    service_description     Swap Usage
      check_command                     check_local_swap!20!10
    }
# Define a service to check SSH on the local machine.
# Disable notifications for this service by default, as not all users may have SSH enabled.
define service{
    use                     local-service       ; Name of service template to use
    host_name               localhost
    service_description     SSH
      check_command                     check_ssh
      notifications_enabled             0
    }
# Define a service to check HTTP on the local machine.
# Disable notifications for this service by default, as not all users may have HTTP enabled.
define service{

  use                   local-service       ; Name of service template to use
    host_name           localhost
    service_description     HTTP
      check_command                     check_http
      notifications_enabled             0
    }
```

6. Contenu de fichier NSC.INI

```
[modules]
NRPEListener.dll
NSClientListener.dll
NSCAAgent.dll
CheckWMI.dll
FileLogger.dll
CheckSystem.dll
CheckDisk.dll
CheckEventLog.dll
CheckHelpers.dll
;# NSCLIENT++ MODULES
;# A list with DLLs to load at startup.
;  You will need to enable some of these for NSClient++ to work.
;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
;*                                    *
;*NOTICE!!!-YOU HAVE TO EDIT THIS*
;*                                    *
;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
FileLogger.dll
CheckSystem.dll
CheckDisk.dll
NSClientListener.dll
NRPEListener.dll
SysTray.dll
CheckEventLog.dll
CheckHelpers.dll
;CheckWMI.dll
; Script to check external scripts and/or internal aliases.
```

CheckExternalScripts.dll

; NSCA Agent if you enable this NSClient++ will talk to NSCA hosts repeatedly (so dont enable unless you want to use NSCA)

NSCAAgent.dll

; LUA script module used to write your own "check deamon".

LUAScript.dll

; RemoteConfiguration IS AN EXTREM EARLY IDEA SO DONT USE FOR PRODUCTION ENVIROMNEMTS!

;RemoteConfiguration.dll

; Check other hosts through NRPE extreme beta and probably a bit dangerous! :)

NRPEClient.dll

; Extreamly early beta of a task-schedule checker

CheckTaskSched.dll

[Settings]

;# OBFUSCATED PASSWORD

; This is the same as the password option but here you can store the password in an obfuscated manner.

; *NOTICE* obfuscation is *NOT* the same as encryption, someone with access to this file can still figure out the

; password. Its just a bit harder to do it at first glance.

;obfuscated_password=Jw0KAUUdXlAAUwASDAAB

;# PASSWORD

; This is the password (-s) that is required to access NSClient remotely. If you leave this blank everyone will be able to access the daemon remotly.

password=##########

;# ALLOWED HOST ADDRESSES

; This is a comma-delimited list of IP address of hosts that are allowed to talk to the all daemons.

; If leave this blank anyone can access the deamon remotly (NSClient still requires a valid password).

; The syntax is host or ip/mask so 192.168.0.0/24 will allow anyone on that subnet access

allowed_hosts=10.4.43.20/24

```
;# USE THIS FILE

; Use the INI file as opposed to the registry if this is 0 and the use_reg in the registry is set to
1

; the registry will be used instead.

use_file=1

allowed_hosts=10.4.43.20

password=ci2010

; # USE SHARED MEMORY CHANNELS

; This is the "new" way for using the system tray based on an IPC framework on top shared
memmory channels and events.

; It is brand new and (probably has bugs) so dont enable this unless for testing!

; If set to 1 shared channels will be created and system tray icons created and such and such...

;shared_session=0

[log]

;# LOG DEBUG

; Set to 1 if you want debug message printed in the log file (debug messages are always
printed to stdout when run with -test)

debug=1

;# LOG FILE

; The file to print log statements to

file=nsclient.log

;# LOG DATE MASK

; The format to for the date/time part of the log entry written to file.

;date_mask=%Y-%m-%d %H:%M:%S

;# LOG ROOT FOLDER

; The root folder to use for logging.

; exe = the folder where the executable is located

; local-app-data = local application data (probably a better choice then the old default)

root_folder=exe

[NSClient]
```

;# ALLOWED HOST ADDRESSES

; This is a comma-delimited list of IP address of hosts that are allowed to talk to NSClient deamon.

; If you leave this blank the global version will be used instead.

allowed_hosts=10.4.43.20

;# NSCLIENT PORT NUMBER

; This is the port the NSClientListener.dll will listen to.

port=12489

;# BIND TO ADDRESS

; Allows you to bind server to a specific local address. This has to be a dotted ip adress not a hostname.

; Leaving this blank will bind to all avalible IP adresses.

bind_to_address=10.4.43.20

;# SOCKET TIMEOUT

; Timeout when reading packets on incoming sockets. If the data has not arrived withint this time we will bail out.

socket_timeout=30

[NRPE]

;# NRPE PORT NUMBER

; This is the port the NRPEListener.dll will listen to.

;port=5666

;# COMMAND TIMEOUT

; This specifies the maximum number of seconds that the NRPE daemon will allow plug-ins to finish executing before killing them off.

;command_timeout=60

;# COMMAND ARGUMENT PROCESSING

; This option determines whether or not the NRPE daemon will allow clients to specify arguments to commands that are executed.

;allow_arguments=0

;# COMMAND ALLOW NASTY META CHARS

; This option determines whether or not the NRPE daemon will allow clients to specify nasty (as in |`&><'"\[]{}) characters in arguments.

```
;allow_nasty_meta_chars=0

;# USE SSL SOCKET

; This option controls if SSL should be used on the socket.

;use_ssl=1

;# BIND TO ADDRESS

; Allows you to bind server to a specific local address. This has to be a dotted ip adress not a
hostname.

; Leaving this blank will bind to all avalible IP adresses.

; bind_to_address=

;# ALLOWED HOST ADDRESSES

; This is a comma-delimited list of IP address of hosts that are allowed to talk to NRPE
deamon.

; If you leave this blank the global version will be used instead.

;allowed_hosts=

;# SCRIPT DIRECTORY

; All files in this directory will become check commands.

; *WARNING* This is undoubtedly dangerous so use with care!

;script_dir=scripts\

;# SOCKET TIMEOUT

; Timeout when reading packets on incoming sockets. If the data has not arrived withint this
time we will bail out.

;socket_timeout=30

[Check System]

;# CPU BUFFER SIZE

; Can be anything ranging from 1s (for 1 second) to 10w for 10 weeks. Notice that a larger
buffer will waste memory

; so don't use a larger buffer then you need (ie. the longest check you do +1).

;CPUBufferSize=1h

;# CHECK RESOLUTION

; The resolution to check values (currently only CPU).
```

```
; The value is entered in 1/10:th of a second and the default is 10 (which means ones every second)

;CheckResolution=10

;# CHECK ALL SERVICES

; Configure how to check services when a CheckAll is performed.

; ...=started means services in that class *has* to be running.

; ...=stopped means services in that class has to be stopped.

; ...=ignored means services in this class will be ignored.

;check_all_services[SERVICE_BOOT_START]=ignored

;check_all_services[SERVICE_SYSTEM_START]=ignored

;check_all_services[SERVICE_AUTO_START]=started

;check_all_services[SERVICE_DEMAND_START]=ignored

;check_all_services[SERVICE_DISABLED]=stopped

[External Script]

;# COMMAND TIMEOUT

; This specifies the maximum number of seconds that the NRPE daemon will allow plug-ins to finish executing before killing them off.

;command_timeout=60

;# COMMAND ARGUMENT PROCESSING

; This option determines whether or not the NRPE daemon will allow clients to specify arguments to commands that are executed.

;allow_arguments=0

;# COMMAND ALLOW NASTY META CHARS

; This option determines whether or not the NRPE daemon will allow clients to specify nasty (as in |`&><'"\[]{}) characters in arguments.

;allow_nasty_meta_chars=0

;# SCRIPT DIRECTORY

; All files in this directory will become check commands.

; *WARNING* This is undoubtedly dangerous so use with care!

;script_dir=c:\my\script\dir

[Script Wrappings]
```

```
vbs=cscript.exe //T:30 //NoLogo scripts\lib\wrapper.vbs %SCRIPT% %ARGS%

ps1=cmd /c echo scripts\%SCRIPT% %ARGS%; exit($lastexitcode) | powershell.exe -
command -

bat=scripts\%SCRIPT% %ARGS%

[External Scripts]

;check_es_long=scripts\long.bat

;check_es_ok=scripts\ok.bat

;check_es_nok=scripts\nok.bat

;check_vbs_sample=cscript.exe //T:30 //NoLogo scripts\check_vb.vbs

;check_powershell_warn=cmd /c echo scripts\powershell.ps1 | powershell.exe -command -

[External Alias]

alias_cpu=checkCPU warn=80 crit=90 time=5m time=1m time=30s

alias_cpu_ex=checkCPU warn=$ARG1$ crit=$ARG2$ time=5m time=1m time=30s

alias_disk=CheckDriveSize MinWarn=10% MinCrit=5% CheckAll FilterType=FIXED

alias_service=checkServiceState CheckAll

alias_process=checkProcState $ARG1$=started

alias_mem=checkMem MaxWarn=80% MaxCrit=90% ShowAll type=physical

alias_up=checkUpTime MinWarn=1d MinWarn=1h

alias_file_age=checkFile2 filter=out "file=$ARG1$" filter-written=>1d MaxWarn=1
MaxCrit=1 "syntax=%filename% %write%"

alias_file_size=checkFile2 filter=out "file=$ARG1$" filter-size=>$ARG2$ MaxWarn=1
MaxCrit=1 "syntax=%filename% %size%"

alias_file_size_in_dir=checkFile2 filter=out pattern=*.txt "file=$ARG1$" filter-
size=>$ARG2$ MaxWarn=1 MaxCrit=1 "syntax=%filename% %size%"

alias_event_log_old=CheckEventLog file=application file=system filter=new filter=out
MaxWarn=1 MaxCrit=1 filter-generated=>2d filter-severity==success filter-
severity==informational truncate=800 unique descriptions "syntax=%severity%: %source%:
%message% (%count%)"

alias_event_log_new=CheckEventLog file=application file=system MaxWarn=1 MaxCrit=1
"filter=generated gt -2d AND severity NOT IN ('success', 'informational')" truncate=800
unique descriptions "syntax=%severity%: %source%: %message% (%count%)"

alias_event_log=alias_event_log_new

check_ok=CheckOK Everything is fine!
```

```
[Wrapped Scripts]
;check_test_vbs=check_test.vbs /arg1:1 /arg2:1 /variable:1
;check_test_ps1=check_test.ps1 arg1 arg2
;check_test_bat=check_test.bat arg1 arg2
;check_battery=check_battery.vbs
;check_printer=check_printer.vbs
; [includes]
;# The order when used is "reversed" thus the last included file will be "first"
;# Included files can include other files (be carefull only do basic recursive checking)
; myotherfile.ini
; real.ini
[NSCA Agent]
;# CHECK INTERVALL (in seconds)
;  How often we should run the checks and submit the results.
;interval=5
;# ENCRYPTION METHOD
;  This option determines the method by which the send_nsca client will encrypt the packets it sends
;  to the nsca daemon. The encryption method you choose will be a balance between security and
;  performance, as strong encryption methods consume more processor resources.
;  You should evaluate your security needs when choosing an encryption method.
; Note: The encryption method you specify here must match the decryption method the nsca daemon uses
;      (as specified in the nsca.cfg file)!!
; Values:
;         0 = None        (Do NOT use this option)
;         1 = Simple XOR  (No security, just obfuscation, but very fast)
;  2 = DES
;  3 = 3DES (Triple DES)
```

```
;       4 = CAST-128
;       6 = xTEA
;       8 = BLOWFISH
;       9 = TWOFISH
;       11 = RC2
;       14 = RIJNDAEL-128 (AES)
;       20 = SERPENT
;encryption_method=14
;# ENCRYPTION PASSWORD
; This is the password/passphrase that should be used to encrypt the sent packets.
;password=
;# BIND TO ADDRESS
; Allows you to bind server to a specific local address. This has to be a dotted ip adress not a hostname.
; Leaving this blank will bind to "one" local interface.
; -- not supported as of now --
;bind_to_address=
;# LOCAL HOST NAME
; The name of this host (if empty "computername" will be used.
;hostname=
;# NAGIOS SERVER ADDRESS
; The address to the nagios server to submit results to.
;nsca_host=192.168.0.1
;# NAGIOS SERVER PORT
; The port to the nagios server to submit results to.
;nsca_port=5667
;# CHECK COMMAND LIST
; The checks to run everytime we submit results back to nagios
; Any command(alias/key) starting with a host_ is sent as HOST_COMMAND others are sent as SERVICE_COMMANDS
```

```
; where the alias/key is used as service name.
[NSCA Commands]
;my_cpu_check=checkCPU warn=80 crit=90 time=20m time=10s time=4
;my_mem_check=checkMem MaxWarn=80% MaxCrit=90% ShowAll type=page
;my_svc_check=checkServiceState CheckAll exclude=wampmysqld exclude=MpfService
;host_check=check_ok
;# REMOTE NRPE PROXY COMMANDS
; A list of commands that check other hosts.
; Used by the NRPECLient module
[NRPE Client Handlers]
;check_other=-H 192.168.0.1 -p 5666 -c remote_command -a arguments
;# LUA SCRIPT SECTION
; A list of all Lua scripts to load.
;[LUA Scripts]
;scripts\test.lua
```