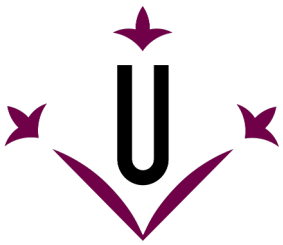

Universitat de Lleida
Escola Politècnica Superior

Màster en Enginyeria de Programari Lliure

TREBALL DE FINAL DE MÀSTER

Implantació del sistema Sauron per a la gestió del sistema DNS de la UdL

Autor:
Gerard Bosch Monserrate

Director:
Dr. Carles Mateu Piñol

26 de setembre de 2011

Agraïments

Voldria agrair a la gent de la divisió de Comunicacions i Sistemes de l'Àrea de Sistemes i Comunicacions de la Universitat de Lleida, bons companys; i en especial al Fermín, qui va encomanar-me la tasca que presenta aquest treball i qui va donar-me suport i consell tècnic des del primer dia.

També agrair al Carles Mateu, qui va suggerir-me aquest tema per abordar com a Treball de Fi de Màster i alhora va acceptar-me com a projectista.

En darrer lloc agrair als qui van dedicar algun minut a respondre alguna consulta a la llista de correu de Sauron.

Llicència

Resum

El sistema de noms de domini (DNS) proveeix d'un sistema distribuït per a la resolució de noms de host en la infraestructura d'internet. Aquest sistema permet que cada organització gestioni les dades dels noms dels seus nodes en la jerarquia del sistema DNS. En una organització però, es poden donar diversos nivells de delegació a determinades parts de la infraestructura de l'organització. Per tal de descentralitzar la gestió d'aquestes parts es presenta un sistema de programari lliure que permet la delegació d'una forma controlada amb control i nivells d'accés d'usuari de forma concurrent i remota a través d'una interfície web. La implementació, configuració, integració i desplegament d'aquest sistema en la xarxa de la Universitat de Lleida es descriu al llarg d'aquesta memòria.

Preàmbul

La motivació del present treball va sorgir de la necessitat d'un sistema que permeti delegar les tasques d'administració dels *hosts* en el servidors de noms de la Universitat de Lleida.

Aquesta memòria està àmpliament centrada en els apèndix tècnics A–D on es descriu la implementació, configuració, integració i desplegament del sistema Sauron en la infraestructura tecnològica de la universitat. Aquests apèndix s'han redactat en anglès a fi de col·laborar en el projecte Sauron amb documentació detallada i actualitzada així com a qualsevol administrador que desitgi migrar la gestió dels seus servidors de noms.

A continuació es descriu l'estructura dels capítols que formen part d'aquest treball.

El capítol 1 introdueix al lector en el problema i objectius a abordar.

El capítol 2 presenta el sistema de noms de domini (DNS) i la tecnologia que hi ha per sota del mateix.

El capítol 3 presenta el protocol DHCP, en dóna les bases teòriques i en descriu el funcionament general.

El capítol 4 descriu la xarxa de la Universitat de Lleida i mostra com integrar el sistema Sauron per a la millora de la gestió dels servidors DNS i DHCP.

El capítol 5 introdueix al lector al sistema de programari lliure Sauron. En dóna una descripció general del funcionament i arquitectura i en realitza un anàlisi a nivell de projecte FLOSS.

El capítol 6 presenta les bases del *backend* dinàmic del servidor de noms BIND i les utilitats per fer ús d'un directori LDAP com a *backend*.

El capítol 7 presenta l'entorn de simulació i proves configurat per a la realització d'aquest treball.

El capítol 8 mostra com s'ha realitzat la integració i desplegament dels diversos components que formen part del sistema en la infraestructura de la Universitat de Lleida.

El capítol 9 presenta les conclusions extretes al final del desenvolupament d'aquest treball així com les vies obertes per al treball futur.

L'apèndix A descriu en detall la instal·lació i desplegament del sistema Sauron.

L'apèndix B descriu la configuració i desplegament del *backend* dinàmic per al servidor BIND fent ús de OpenLDAP.

L'apèndix C llista els pegats de codi que s'han generat per a les correccions i modificacions realitzades sobre el sistema Sauron i que són d'aplicació en l'apèndix A.

L'apèndix D llista els fitxers LDIF (LDAP Interchange Format) utilitzats en l'apèndix B.

Índex

Índex de figures

Índex de taules

Listings

Índex d'enllaços a fitxers

Capítol 1

Introducció

Un dels recursos crítics en el funcionament d'una xarxa de comunicacions i en especial de la xarxa d'Internet rau en la resolució de noms dels nodes que integren la xarxa. El sistema DNS proporciona un mecanisme distribuït per a la resolució de noms que permet la delegació de la gestió de les parts del sistema. Aquest fou el problema principal abans del desenvolupament d'aquest sistema: la centralització de la gestió. Amb l'especificació del sistema DNS aquest problema es va solucionar aportant un nou sistema distribuït.

Amb el sistema DNS un organisme es pot encarregar de la gestió de noms dels seus nodes de forma autònoma i que les dades que gestiona siguin visibles des de tota la infraestructura d'Internet. Per tant cada organisme haurà de mantenir una forma d'organització de les dades del sistema de noms.

1.1 Motivacions i objectius

Aquesta organització però, es pot convertir en una tasca complexa quan l'envergadura de l'entitat és prou important degut al nombre de nodes i nivells que cal que gestioni. A més a més, la implementació del servidor de noms *de facto* (ISC BIND), maneja aquestes dades través de fitxers de configuració que tendeixen a crèixer i tornar-se difícils de mantenir.

Si pensem en la delegació de permisos dins d'un organisme, aquest sistema manual de manteniment basat en fitxers tampoc resulta de gran utilitat. En el cas de la Universitat de Lleida (UdL), es manté un servei de noms extern connectat al servei de noms global d'Internet i alhora un servei de noms intern utilitzat per a la gestió dels equips interns. A més a més, l'estructura organitzativa dels campus universitaris implica que diferent personal d'administració i suport informàtic s'hagi de fer càrrec de diferents parts i *hosts* de la xarxa. Així doncs, cal algun mecanisme que permeti

la descentralització, ara però, del manteniment d'aquestes parts de l'organisme, tot delegant permisos i permetent restringir-ne els nivells d'accés.

El sistema que es presenta, aporta aquest component de descentralització i delegació de drets i a més a més abstrau l'administrador dels fitxers de dades, que com es comenta tendeixen a créixer i a convertir-se, en algunes ocasions, en caòtics i de difícil manteniment; proporcionant una interfície amigable per a la gestió que permetrà agilitzar el manteniment.

Per tal d'assolir aquest objectiu es proposa de la implantació del sistema de programari lliure Sauron, el qual permetrà delegar aquestes tasques de gestió d'una forma restringida a través d'una interfície d'usuari web. També es presentaran algunes modificacions en l'arquitectura actual del sistema DNS que s'integraran amb l'ús del sistema Sauron per tal de realitzar una millora en l'organització d'aquestes dades, així com una sèrie de modificacions, correccions i millores del programari Sauron presentades en forma de pegats de codi.

Finalment es descriurà el desplegament global del sistema, descrivint cadascuna de les parts que el formen i les solucions desenvolupades per tal de dur a terme la integració i desplegament en el context de la Universitat de Lleida.

Capítol 2

DNS. Introducció tecnològica

El servei de resolució de noms de domini d'Internet, conegut com a sistema de noms de domini (*Domain Name System*), abreujat DNS, proporciona un sistema global distribuït de resolució de noms. Aquest sistema està distribuït en la xarxa d'Internet i és un dels recursos crítics en funcionament d'aquesta xarxa. En aquest capítol es presenta la tecnologia que hi ha per sota del sistema DNS així com el seu funcionament.

El sistema DNS permet traduïr el nom descriptiu d'un recurs a la seva adreça IP. Els servidors DNS juguen un paper clau en el funcionament d'Internet: proveeixen d'una distribució de recursos de xarxa (hosts) emprant noms descriptius enlloc d'adreces IP que resultarien no només poc pràctic sinó impossibles de recordar des del punt de vista de l'usuari.

A més a més, proporciona un nivell d'indirecció que permet que la reconfiguració de recursos de xarxa no afecti a la seva localització (e.g. si un host canvia d'adreça, el sistema de noms de domini permet mantenir-ne el mateix nom). D'aquesta manera fa que les xarxes ofereixin millor resposta als canvis.

2.1 Antecedents

Davant la capacitat limitada que tenim les persones de recordar llargues seqüències numèriques com són les adreces de xarxa, és indispensable disposar d'un sistema capaç de traduïr aquestes adreces a noms descriptius que en facin més senzilla la tasca de memoritzar-los i alhora d'identificar a qui corresponen.

El sistema de noms de domini (en anglès *Domain Name System*), en endavant DNS, nasqué de la mà d'ARPAnet[1]. Durant els anys 70, la xarxa ARPAnet feia ús d'un

[1] ARPAnet va ser una xarxa experimental d'ordinadors fundada a finals dels anys 60 per l'Agència de Projectes Avançats de Recerca del Departament de Defensa dels EUA que connectava diverses organitzacions de recerca i que fou la precursora de l'actual Internet

fitxer de *hosts* (HOSTS.TXT) que contenia un mapa de noms i adreces de cadascun dels nodes connectats a la xarxa, similar al fitxer de hosts de UNIX. Aquest fitxer era mantingut i distribuït des d'un únic punt ubicat a l'Institut de Recerca de Stanford i les actualitzacions sobre el fitxer s'enviaven per FTP de forma setmanal. A mesura que la xarxa anava creixent el fitxer de hosts també ho feia i el sistema de distribució d'aquest es va convertir en inviable degut no només a la mida del fitxer en si, sinó al trànsit que es generava en el procés d'actualització.

Amb l'estandardització de la pila de protocols TCP/IP a principis dels anys 80, ARPAnet va realitzar una migració a aquest protocol i el nombre de *hosts* de la xarxa es va disparar. Això va fer que el sistema de noms basat en el el fitxer de hosts presentés els següents problemes:

- *Trànsit i càrrega*: La distribució del fitxer des d'un únic node era insostenible des del punt de vista tant del transit que s'hi generava com de la càrrega de CPU.
- *Col·lisió de noms*: No existia cap mecanisme autoritari per determinar els noms dels hosts i per tant res no prevenia que es poguéssin repetir, trencant d'aquesta forma l'esquema de la traducció de les adreces.
- *Consitència*: El fitxer de hosts no podia reflectir els canvis en la xarxa d'una forma eficaç.

Resumint, es tractava d'un problema d'escalabilitat. Les sol·lucions a aquests problemes es tradueixen en:

- la *descentralització* de l'administració del sistema de noms, per tal de d'eliminar els colls d'ampolla;
- la *delegació* per facilitar-ne la gestió;
- l'ús d'un *espai de noms jeràrquic* per tal d'assegurar la unicitat dels noms.

D'aquesta forma a principis dels anys 80, es va dissenyar l'arquitectura del nou sistema de noms anomenat *Domain Name Sistem* (DNS), proporcionant una administració local i mantenint un accés global a les dades. Aquest sistema proporciona un servei global distribuït de resolució de noms. L'especificació definitiva d'aquest nou sistema tal i com es coneix a dia d'avui es va publicar el 1987 als RFCs [Moc87a] i [Moc87b].

2.2 Estructura

L'estructura del sistema de noms de domini està organitzada de forma jeràrquica en un arbre, similar a l'estructura del sistema de fitxers UNIX. Els elements d'aquesta jerarquia són unes etiquetes (o noms) connectades per arestes. En cada nivell d'aquest

arbre una etiqueta ha de ser única per tal que no es produeixin col·lisions entre noms. Les arestes d'aquest arbre es representen per punts (.), per tant el punt és l'element que connecta les etiquetes. Sovint es representa el node arrel d'aquest arbre amb un punt simple quan es vol fer referència a l'arrel. La figura 2.1 mostra un exemple d'aquesta estructura arborescent.

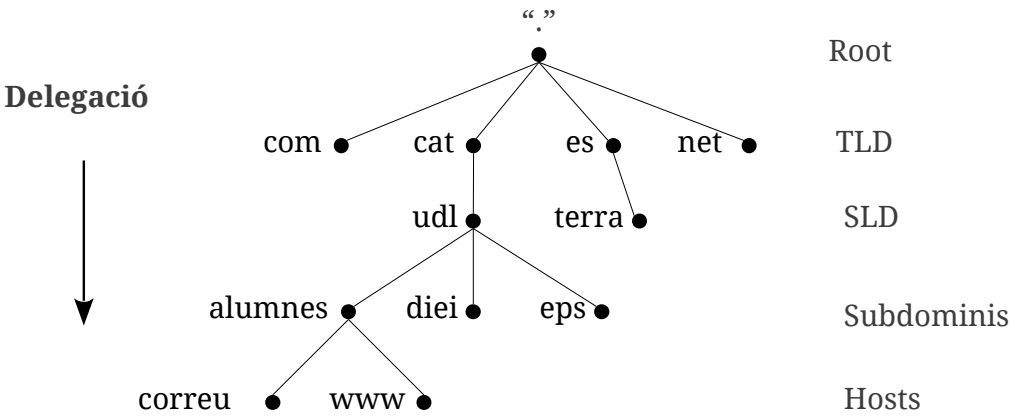

Figura 2.1: Jerarquia del sistema de noms de domini (DNS)

Les definicions que es donen a continuació s'utilitzen sovint de forma equivalent tot i fer referència a conceptes diferents. Per aquest motiu se'n dona la definició i matisa les diferències:

Definició (Domini). Es tracta d'un sub-arbre de la jerarquia de noms.

Definició (Zona). Conjunt de noms que pertanyen a la mateixa unitat, agrupats en el mateix espai de noms.

Mentre que el domini fa referència a tots els nodes de la jerarquia que representa, els nodes d'una zona depenen de la delegació. Així per exemple la zona *udl.cat* conté tots els noms de domini dels nivells inferiors ja que n'és *autoritativa*, mentre que la zona *cat* no conté dades del domini *udl.cat* ja que aquest és delegat. També es pot entendre una zona com una part del domini que és gestionada per un servidor DNS.

Definició (Nom qualificat – FQDN). Es tracta d'un nom de domini absolut, respecte a l'arrel de la jerarquia. Identifica per tant un node de forma unívoca dins de l'espai de noms. Els noms qualificats (*o Fully Qualified Domain Name*) han d'acabar en un punt (.). D'altra manera el sistema DNS els considerarà noms no qualificats i per tant relatius.

Per tal de determinar el nom d'un host, es llegeixen les etiquetes de l'arbre en ordre invers fins arribar al node arrel. El format d'aquestes etiquetes es defineix en

[Moc87b] —tot i que s'han produït algunes modificacions en les regles— i poden estar formades pels caràcters {a-z, A-Z, 0-9, '-'} amb algunes restriccions:

- la llargada màxima d'una etiqueta és de 63 caràcters,
- una etiqueta no pot començar ni acabar amb un guió '-',
- inicialment tampoc es permetia que comencessin amb un número, però aquesta restricció es va eliminar en el RFC1123.

El sistema de noms no distingeix entre majúscules i minúscules, per tant, tenen el mateix significat els noms «udl.cat», «UDL.CAT» i «uDl.CaT».

En el primer nivell de la jerarquia (nivell immediatament inferior a l'arrel) s'hi troben un conjunt d'etiquetes especials anomenades TLD (*Top Level Domains*), els noms de les quals venen determinats per la ICANN[2]. Originalment es van definir set TLDs, destinats a les organitzacions que es mostren en la taula 2.1.

Taula 2.1: Llistat original dels TLDs

TLD	**Ús**
arpa	*Address and Routing Parameter Area*, llegat d'ARPAnet i reservat per a la infraestructura d'Internet.
com	Organitzacions comercials
edu	Organitzacions educacionals (e.g. universitats)
gov	Organitzacions governamentals
mil	Organitzacions militars
net	Organitzacions de xarxa
org	Organitzacions no comercials (e.g. fundacions)
int	Organitzacions internacionals

Els dominis d'aquesta organització inicial es van anomenar posteriorment gTLD (*generic Top Level Domains*). Per tal d'acomodar la internalització d'Internet, es van crear TLDs específics per a cada país (ccTLD), corresponents al codi internacional del país —conforme a l'estàndard ISO 3166. Així cada país podria sol·licitar la creació del seu domini corresponent.

Alguns dominis com *co.uk* o *com.tw* tot i estar formats per dues etiquetes, es consideren TLDs.

El 2004, ICANN va revisar la política dels gTLD per tal de crear un subconjunt anomenat sTLD (*sponsored TLD*). La taula 2.2 llista aquests dominis. Els noms de les etiquetes corresponents als TLD segueixen alguns convenis establerts històricament per tal per tal de prevenir que el sistema de noms esdevingui caòtic.

[2]ICANN (*Internet Corporation for Assigned Names and Numbers*) és una organització nord-americana

Taula 2.2: *Sponsored* TLDs (actualitzat agost de 2011)

TLD	Ús
aero	Indústria aeronàutica
asia	Companyies i organitzacions asiàtiques
cat	Comunitat lingüística catalana
coop	Cooperatives i associacions
edu	Institucions acadèmiques superiors
gov	Govern dels EUA
int	Organitzacions internacionals
jobs	Recursos humans
mil	Exèrcit nord-americà
mobi	Destinat a productes i serveis mòbils
museum	Museus
tel	Informació de contacte
travel	Agències i companyies de turisme i transport
xxx	Entreteniment per adults

Com es veu alguns dominis van estar redefinits com a *sponsored*. També s'han afegit alguns dominis nous com a globals o globals-restringits TLD. Aquests es mostren en la taula 2.3.

Taula 2.3: gTLDs afegits amb posterioritat (actualitzat agost de 2011)

TLD	Ús
biz	Negocis en general
info	Recursos d'informació genèrica
name	Genèric per a persones
pro	Organitzacions professionals

El 1999 en [EP99] es reserven —en acord amb la IANA— de forma permanent alguns TLDs i dominis de sgon nivell (SLD) restringits per a l'ús en proves. Aquests es mostren en la taula 2.4

Per sota dels TLDs s'hi troben els dominis de segon nivell (SLD) i per sota d'aquests s'hi troben subdominis (dominis de tercer nivell) i així successivament, de la forma en que es mostra en la figura 2.1.

sense ànim de lucre, encarregada de la gestió dels TLDs, entre altres. IANA, l'Autoritat de Números d'Internet Assignats, és operada per ICANN.

Taula 2.4: TLD/SLDs reservats per a proves

TLD/SLD	Ús
test	Recomanat per a l'ús en proves de DNS.
example	Recomanat per a l'ús en documentació i exemples.
invalid	Utilitzat per a la construcció de noms de domini que s'espera que siguin invàlids.
localhost	Tradicionalment definit com un registre A que apunta a l'adreça de *loopback*. Un altre ús entraria en conflicte amb les implementacions desplegades del sistema DNS.
example.com *example.net* *example.org*	Exemples

2.2.1 Autoritat i delegació

Cada node de la jerarquia del DNS pertany a una entitat que n'és autoritària, és a dir n'és responsable de l'operació i gestió del node. Es diu que aquesta entitat gestiona el node de forma *autoritativa*. Una entitat pot delegar l'autoritat dels nodes de nivell inferior.

La ICANN és l'autoritat del domini arrel i administra de forma *autoritativa* els TLD, però en delega la gestió a una sèrie d'entitats. En el cas dels ccTLDs cada país defineix les seves regles en quan a la delegació.

Per exemple el domini *cat* és delegat a la Fundació puntCAT, mentre que el domini *udl.cat* és delegat a la Universitat de Lleida. Per tant la UdL és l'entitat autoritària dels nodes inferiors en la jerarquia de noms.

D'aquesta forma s'aconsegueix la descentralització del sistema DNS.

2.2.2 Components del sistema DNS

El sistema DNS està format dels següents components:

- *Servidors DNS*: Contenen i gestionen la informació del domini el qual són autoritatius a través de fitxers de zona o qualsevol altre mitjà de persistència.
- *Resolver*: Programa o llibreria que funciona com a client d'un servidor de noms. Realitza les peticions, n'interpreta les respostes i en retorna la informació al programa que ha realitzat la petició. És habitual que el resolver estigui implementat en una biblioteca del sistema operatiu.

Les dades d'un servidor DNS es defineixen de forma textual en els anomenats registres de recursos, registres DNS —*o RR (Resource Records)*— i són organitzats en zones. El format d'aquests registres i dels fitxers de zona està estandaritzat en [Moc87b], de forma que els fitxers de zona són portables a qualsevol implementació del servidor de noms.

2.2.3 Zones i registres (RR)

La informació d'una zona es troba descrita en els fitxers de zona a través de registres (*Resource Records*), que descriuen les característiques, *hosts* i serveis proporcionats per un domini. Una mala configuració pot provocar que un domini deixi de ser accessible, que el correu electrònic s'enviï a un destí incorrecte,... i una llarga llista de problemes. A més a més, les respostes errònies d'un servidor mal configurat són *cachejades* per altres servidors durant hores, setmanes, dies,... depenent de la configuració del mateix. De manera que un simple error pot interrompre un servei durant un llarg període.

Al llarg dels 30 anys de l'especificació DNS, s'hi han definint algunes extensions i nous registres i, actualment, n'existeixen 72. Una descripció exhaustiva d'aquests queda fora de l'àmbit d'aquest treball i es pot consultar junt als corresponents RFCs al següent enllaç de la IANA. Se'n comentaran però alguns que es consideren d'especial importància o bé interessants per alguna qüestió.

http://www.iana.org/assignments/dns-parameters

El *mapeig directe*, proporciona la resolució directa i defineix les característiques de la zona i les adreces IP utilitzades per qualsevol *host* o servei. De forma contrària, el *mapeig invers* proporciona la resolució inversa i defineix una relació entre les adreces IP i els noms del domini. Aquest mapeig es defineix en els fitxers de zona.

Nota:

Quan es parli d'una zona, sense especificar, s'està fent referència a la zona directa.

L'estructura general d'una zona és la que segueix:

- Dades descriptives de la zona, conegudes com a registre *Start Of Authority* (SOA).
- Dades globals de la zona, com els servidors *autoritatius* de la zona (NS) o els servidors de correu (MX).
- Hosts de la zona, definits a través del registre *Address* (A) o *Pointer* (PTR).
- En el cas de delegació de subdominis,

  * el servidor de noms responsable del subdomini (registre NS),
  * un registre anomenat *glue record* per tal que el servidor pugui arribar al servidor del subdomini (registre A).

A continuació es llisten les directives i registres més habituals que formen part d'una zona i de l'operació bàsica del DNS.

Directives

- `$TTL`: Defineix el temps de vida (*time-to-live*) d'una zona, i.e. el temps que un RR pot ser *cachejat* per un altre servidor DNS. El valor TTL s'utilitza per defecte per a tots aquells registres que no defineixen un TTL específicament. És obligatoria.
- `$ORIGIN`: Es tracta del nom del domini de la zona que es defineix i cal que sigui un nom qualificat (FQDN). Aquest valor es concatena als noms no-qualificats de la zona aplicant així una regla de substitució. És opcional.

Registres

- *SOA*: *Start Of Authority*, descriu les característiques globals de la zona. Cal que sigui el primer registre del fitxer de zona i ha de ser únic. Aquest registre és obligatori.
- *NS*: *Name Server*, defineix els servidors de noms que són *autoritatius* per la zona. Cal especificar-ne almenys dos tal i com es comenta a 2.3.3.
- *MX*: *Mail eXchanger*, defineix els servidors de correu per al domini. És opcional i depén de si el domini gestiona correu electrònic o no.
- *A*: *Address*, s'utilitza per definir les adreces IPv4 dels *hosts* de la zona. Per definir entrades IPv6 el registre és AAAA.
- *CNAME*: *Canonical Name*, defineix un àlies. És opcional.

Nota:

Qualsevol nom no qualificat que aparegui en els fitxers de zona serà substituït aplicant la regla de substitució de ORIGIN. Això és concatenar el valor `$ORIGIN` al valor del nom no-qualificat.

Compte!

Quan s'especifiquin noms de domini absoluts, cal que aquests acabin en punt '.' (notació FQDN). En cas contrari es realitzarà la substitució per $ORIGIN, provocant resultats indesitjats.

El registre SOA

Com ja s'ha avançat aquest registre conté informació important que afecta a tota la zona. La definició d'aquest registre és la següent:

`name ttl class SOA name-server email-addr (sn refresh retry expiry negative-ttl),`

i la descripció de cada camp es mostra en la taula 2.5. El separador de camps és l'espai en blanc o tabulador (diversos espais s'interpreten com un de sol). A continuació se'n mostra un exemple per a la zona example.com:

```
@    IN    SOA    ns.example.com. hostmaster.example.com. (
                 2003080800 ; sn = serial number
                 172800     ; refresh = refresh = 2d
                 900        ; retry = update retry = 15m
                 1209600    ; expiry = expiry = 2w
                 3600       ; negative-ttl = 1h
                 )
```

El registre MX

Els registres MX defineixen els servidors que processen el correu enviat a un domini. Aquests registres tradueixen el nom del domini d'una adreça de correu electrònic al nom del servidor que l'ha de processar. Quan un usuari envia un correu electrònic, el valor registre MX de la consulta DNS és el que s'utilitza per determinar a quina adreça IP cal enviar el correu.

La definició formal d'aques registre és la que segueix,

`name ttl class MX preference name`, on,

- els camps `ttl` i `class`, a l'igual que en el registre SOA i la resta de registres especifiquen de forma opcional el temps de vida del registre (i.e. el temps que un registre pot ser *cachejat*), i la classe Internet.

  En el cas dels registres MX és habitual establir un TTL major que el temps per defecte de la zona, donat que rarament el nom d'un *relay* de correu canviarà encara que el seu corresponent registre A canvii. D'aquesta manera es redueix la càrrega en el DNS, al reduïr el nombre de consultes que aquest rep per peticions de registres MX, d'altra banda molt utilitzats.

- el camp `preference` és un nombre que indica la prioritat del registre. Com menor és el seu valor, major és la prioritat. Típicament existiran diversos registres MX amb diferents prioritats.

Taula 2.5: Descipció exemplificada del registre SOA

Camp	**Valor**
name	@
	Nom qualificat de la zona (@ té el valor origin)
ttl	
	És opcional en tots els registres. Al no estar especificat, s'utilitza el valor definit per la directiva `$TTL`.
class	IN
	Defineix la classe dels registres com Internet. Existeixen altres valor però són rarament utilitzats.
name-server	ns.example.com.
	Defineix el nom del servidor primari de la zona. Té un significat especial quan s'utilitza en configuracions *Dynamic DNS*. Cal que aquest nom es defineixi igualment utilitzant un registre NS. En l'argot DNS s'anomena camp MNAME.
email-addr	hostmaster.example.com.
	Defineix l'adreça de correu de l'administrador de la zona. La recomanació del RFC 2142 és que s'utilitzi l'adreça *hostmaster@domini*. Donat que el caràcter '@' té un altre significat en el context de DNS, s'utilitzen punts per separar els camps. El primer punt es correspon al simbol arroba.
SN	2011080800
	Defineix el nombre de sèrie associat a la zona. Aquest nombre ha de ser incrementat cada cop que es produeix un canvi en el fitxer de zona. És un valor de 32 bits sense signe, per tant pren valors entre 0 i 4294967295. Per conveni s'utilitza la data en format AAAAMMDDrr, on 'rr' és el nombre de revisió dins d'un dia. Aquest valor és utilitzat en les transferències de zona i és molt important que mantingui la consistència.
refresh	2d
	Quan aquest temps expira, els servidors esclaus comproven si s'han produït canvis en la zona mestra comprovant el valor SN i per tant si és necessària una transferència de zona.
retry	15m
	Defineix l'interval de reintent en cas que un servidor esclau falli en contactar amb el servidor mestre. Típicament de 10 a 60 minuts.
expiry	2w
	Indica el temps d'expiració dels registres (temps en que un registre deixa de ser autoritatiu). Només fan ús d'aquest valor els servidors esclaus. Els esclaus deixaran de respondre peticions per a la zona quan aquest temps hagi expirat i no s'hagi produït cap contacte amb el servidor mestre. Quan el comptador *refresh* expira, l'esclau contacta amb el mestre i reinicia de nou tots dos comptadors.
negative-ttl	3600
	Període de *cacheig* de respostes negatives. És el temps màxim que una resposta negativa a una petició pot ser *cachejada*.

La prioritat no només s'utilitza per establir servidors de suport en cas que el de prioritat més alta no respongui. El seu ús es pot utilitzar per configurar polítiques *anti-spam*, per exemple, establint un servidor inexistent com a principal.

Per tal de realitzar un balanceig de càrrega entre diferents servidors de correu, es poden configurar diversos registres A, en aquest cas, el servidor de noms BIND realitza una rotació Round-Robin en les respostes, proporcionant així un mecanisme de balanceig.

- el camp `name` indica el nom del domini del servidor de correu precedit per la preferència.

El registre CNAME

Aquests registres s'utilitzen per crear àlies. Aquests àlies es poden utilitzar per nombrar diversos serveis sobre un *host* existent (e.g. FTP, WWW,...). També són d'utilitat per configurar diversos *hostings* virtuals, que correran sobre el mateix *host*.

La seva definició és la següent:

```
name ttl class CNAME canonical-name.
```

El nom *name* serà un àlies del nom que apareix al camp *canonical-name*.

Consideracions en l'ús de registres CNAME:

- Encadenar diversos registres CNAME, no és una bona pràctica i està desaconsellat en els RFCs. Aquest ús podria desencadenar en bucles i a més alenteixen la resolució.
- Un àlies no hauria d'aparèixer mai en les dades —la part dreta— de qualsevol altre registre, utilitzar enlloc el nom canònic.
- No haurien de ser utilitzats mai amb registres MX o NS.
- Un nom apareix a la dreta d'un registre, no ha d'aparèixer a la part esquerra d'un registre CNAME.
- El seu ús provoca més càrrega en el servidor: haurà de resoldre el CNAME i el RR al qual apunta.
- En la resposta a una petició d'un registre CNAME s'inclou tant el registre CNAME com el registre al qual apunta. Això pot causar que la resposta superi els 512 octets de límit del paquet UDP, reduïnt-ne el rendiment.

Ús del CNAME

En general la funcionalitat dels registres CNAME es pot assolir fent ús del registre A. Existeixen però dues situacions en les que l'ús del d'aquest registre és imprescindible:

- el nom canònic resideix en en un domini extern (no es té autoritat sobre el nom canònic),
- l'accés a *hostings* virtuals que corren sobre un únic *host*.
- s'utilitza també en la delegació de mapes inversos per a *subnets* (veieu 2.2.5).

El registre TXT

Històricament utilitzat per definir text genèric associat a un nom. Aquest registre s'ha utilitzat com a mecanisme transaccional per configurar el protocol SPF (*Sender Policy Framework*), per tal d'evitar la recepció de correu *spam* mitjançant la verificació de les fonts legítimes de correu. L'abril de 2006 es va publicar [WS06] on es defineix un nou registre (SPF) dedicat a aquest propòsit però l'ús del registre TXT no està considerat obsolet donat que una bona part del programari de servidor DNS (versions antigues) no suporta aquest registre. La recomanació dels RFC és utilitzar el registre SPF junt al TXT.

Definició:

```
name ttl class TXT text
```

El protocol SPF permet especificar quins *hosts* d'un domini estan autoritzats a enviar correu. El receptor del missatge pot rebutjar-lo si aquest no prové d'una font autoritzada. Usualment es defineix SPF en el domini i els servidors de correu. La funcionalitat del registre SPF és la mateixa que la que s'assoleix amb l'ús del registre TXT.

2.2.4 Mapeig invers i zona inversa

La zona inversa està destinada a manejar la resolució inversa de noms de domini. Això és, donada una adreça IP, obtenir-ne el corresponent nom de domini. Aquest tipus de resolució s'utilitza sovint amb proposits relacionats amb el diagnostic o la seguretat per tal de traçar-ne l'origen.

Històricament el mapeig invers d'adreces IPv4 no ha estat obligatori, tot i això el seu ús és recomanat. Els sistemes de correu actuals, utilitzen aquesta resolució inversa amb el propòsit de realitzar una autenticació senzilla. El mapeig invers fa ús del domini especial *IN-ADDR.ARPA* i dels registres PTR.

El mapeig invers utilitza una lleugera variació de la jerarquia de noms descrita anteriorment. Mentre que en un nom de domini el node superior de la jerarquia es correspon a la darrera etiqueta del nom (e.g. el node superior de *www.example.com* és *com*) el mapeig invers s'organitza de forma contrària. Per tant, considerant l'adreça de classe C *192.168.25.15*, el node superior es correspon a *192* i no pas a *15*.

Per tal d'estructurar la jerarquia, s'utilitza el domini *IN-ADDR.ARPA* com a base i es disposa l'adreça (en l'ordre invers) sota daquest domini. Al tractar-se en aquest exemple d'una classe C, el host s'identifica per l'etiqueta *5* i la zona per la part de l'adreça corresponent a la xarxa (*192.168.25*) i s'utilitza de forma similar a la zona directa mitjançant el registre PTR.

Així, en l'exemple anterior s'obté,

`15.25.168.192.IN-ADDR.ARPA`,

com a nom del domini corresponent a l'adreça IP 192.168.25.15. En aquest cas, la zona 25.168.192.IN-ADDR.ARPA, és *autoritativa* del host *15*, corresponent a l'adreça IP sencera.

Cal tenir en compte però, que la resolució inversa difereix de la directa en el sentit de la delegació. Quan es tracta de *subnets* el mecanisme de delegació és diferent donat que s'ha de permetre una delegació múltiple del domini, que en aquest exemple es correspon a *25.168.192.IN-ADDR.ARPA*. Aquest procediment es veu més endavant.

El registre PTR

Per tal de definir les zones inverses s'utilitza el registre PTR, el qual defineix el mapeig entre una adreça i un nom de domini. La definició d'aquest registre és la següent:

```
name ttl class PTR name
```

La taula 2.6 en descriu l'exemple següent:

```
15        IN    PTR    www.example.com.
```

2.2.5 Delegació de mapes inversos

En l'RFC 2317, es descriu la tècnica per tal de permetre la delegació del mapeig invers de *subnets*. Com ja s'ha vist, en la resolució inversa utilitza l'adreça invertida (la part de l'adreça de xarxa), però què passa quan es vol delegar la gestió d'una subnet (e.g. 192.168.25.0/26)? En aquestes circumstàncies s'obtenen 4 xarxes amb l'identificador 192.168.25, per tant la delegació es pot produïr en quatre entitats diferents on cadascuna ha de poder gestionar els mapes inversos d'acord als seus *hosts*.

Taula 2.6: Exemple descriptiu de la zona inversa

Camp	Valor
name	15 Encara que en aquest cas sigui un número, és tractat com una etiqueta. Al ser un nom no–qualificat, el valor $ORIGIN —en aquest cas *25.168.192.IN-ADDR.ARPA*— serà concatenat, com ja s'ha comentat anteriorment.
ttl	Al no definir el TTL, s'utilitza el valor genèric per la zona.
class	IN Defineix la classe Internet, en l'igual que la resta de registres.
name	example.com. Es tracta del nom de domini associat a l'adreça IP. Cal que aquest nom sigui qualificat o serà substituït pel valor $ORIGIN resultant en un nom incorrecte.

Per tal de dur a terme aquesta delegació, cal l'ús d'una tècnica i notació especial que fa ús dels registres CNAME. Caldrà que tant l'entitat que delega com l'entitat delegada utilitzin aquesta notació en els fitxers de zona per tal d'indicar que es tracta de *subnets* i permetre d'aquesta forma una delegació i gestió múltiple del mateix domini (*25.168.192.IN-ADDR.ARPA*).

2.3 Funcionament

El protocol DNS és un protocol de la capa d'aplicació (nivell 5) del model TCP/IP. Les consultes DNS utilitzen per defecte el protocol UDP per al transport i el port 53. Per raons de rendiment, l'especificació restringeix la mida del paquet UDP a 512 octets. D'altra banda quan una resposta excedeix els 512 octets, s'utilitza el protocol TCP.

Tipicament s'evita excedir els 512 octets en les respostes per evitar l'*overhead* que afegeix l'ús de TCP. Tal i com es veurà, aquest és el motiu pel qual el nombre de *root-servers* es manté en 13.

El funcionament del sistema DNS, es basa en els servidors de noms (*o NS*). Aquests servidors es troben repartits en els diversos nivells de la jerarquia i gestionen la informació dels noms dels dominis els quals són *autoritatius*.

Aquests servidors són els encarregats de realitzar la traducció de noms a adreces IP, a través de les consultes que els clients realitzen. Per tant cada *host* connectat a Internet cal que tingui configurada almenys l'adreça IP d'un servidor de noms per tal que el *host* pugui resoldre els noms de domini. D'altra manera el *host* només podrà comunicar-se amb la resta de *hosts* d'Internet a través d'adreces IP, el qual resulta poc pràctic si darrere hi ha una persona humana.

Els servidors de noms de l'arrel de la jerarquia (*root-servers*) són un dels recursos més crítics d'Internet. Quan un servidor de noms rep una petició per a la resolució d'un nom de domini del qual no disposa d'informació, aquest realitza una petició als *root-servers*, i aquests responen amb una remissió al corresponent TLD, el qual conté informació sobre el següent servidor de la jerarquia del nom i retorna una remissió a aquest. Aquest procediment es repeteix successivament fins arribar a un servidor *autoritatiu* pel domini que s'intenta resoldre. Aquest darrer contindrà la informació sobre el nom que es sol·licita i serà l'encarregat de retornar-la.

Els *root-servers* són responsabilitat de la ICANN i són coneguts per tots els servidors de noms a través de la distribució d'un fitxer especial de zona que tot el programari de servidor DNS duu incorporat. Existeixen 13 els *root-servers* repartits per tot el món i es preveu que aquest nombre no creixi en el futur pròxim, degut a que és el límit que permet que les respostes pugin ser enviades en un únic paquet UDP de 512 octets, reduint la càrrega dels servidors. Aquest límit és part de la pròpia especificació del protocol DNS [Moc87b]. Els 13 *root-servers* disposen del seu propi domini *root-servers.net* i els noms dels mateixos es corresponen de les lletres ≪A≫ a ≪M≫ (e.g. *c.root-servers.net*).

2.3.1 Peticions (*Queries*)

La tasca principal d'un servidor és la de respondre les peticions dels *resolvers*. Els resolvers generen les peticions DNS sol·licitades per l'usuari o aplicació. Per exemple una consulta estàndard és ≪Quina és l'adreça IP de www.example.com?≫. El resolver genera la petició i l'envia al servidor DNS que té configurat (en sistemes *NIX això és el fitxer `/etc/resolv.conf`).

Existeixen tres tipus de peticions DNS, les quals es descriuen a continuació:

1. **Peticions recursives**: En aquest tipus de peticions, el servidor DNS realitzarà tot el treball necessari per tal de resoldre el nom, preguntant als diferents servidors de la jerarquia en cas que sigui necessari fins a resoldre el domini o bé retornar un error en cas que el domini no existeixi (NXDOMAIN) o no es pugui accedir a algun servidor degut a problemes en la xarxa, etc. En les respostes s'indica si les dades són *autoritatives* o *cachejades*. Un servidor DNS no te perquè resoldre peticions recursives.

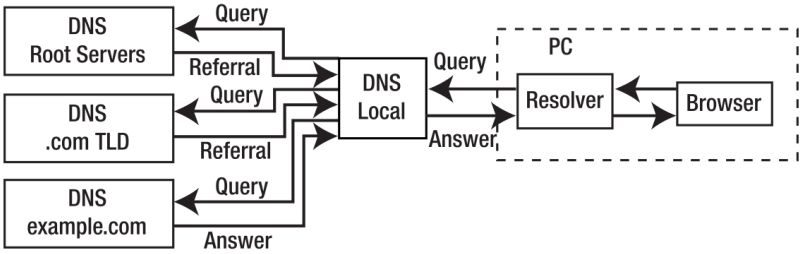

Figura 2.2: Diagrama d'una petició recursiva.

2. **Peticions iteratives**: En les peticions iteratives el servidor respondrà la petició si disposa de la informació sol·licitada. En cas contrari respondrà parcialment amb la informació que sigui d'utilitat a la petició realitzada (un *referral* o remissió a un altre servidor) però sense realitzar cap consulta addicional a cap altre servidor DNS, o un error. Serà aleshores responsabilitat del *resolver* la resolució completa del nom. Els *root-servers* i els servidors dels TLDs només responen peticions iteratives (no recursives): aquests no poden malgastar recursos degut a l'elevada càrrega que suporten. En les respostes s'indica si les dades són *autoritatives* o *cachejades*. Tots els servidors DNS han de suportar les peticions iteratives.

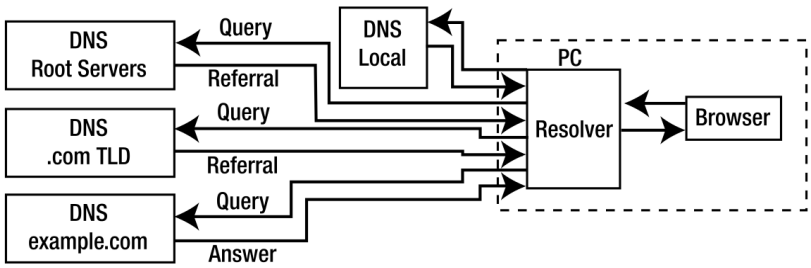

Figura 2.3: Diagrama d'una petició iterativa.

Nota: *Els resolvers instal·lats en el sistema operatiu del client consisteixen habitualment en una llibreria i són anomenats* stub resolvers. *Aquests són incapaços de manejar* referrals. *Per tant el servidor DNS que el client tingui configurat haurà de suportar peticions recursives per tal de manejar els* referrals. *Aquest fet no s'il·lustra en la figura 2.3.*

---

3. **Peticions inverses**: Aquest tipus de peticions consisteix en l'obtenció d'un nom de domini IP a partir d'un registre. Aquest tipus de peticions ha estat rarament implementat i va quedar obsolet en [Law02] en favor de l'ús del *mapeig invers* i els registres PTR on es fa ús de peticions iteratives (no recursives) amb el domini especial *IN-ADDR.ARPA*.

2.3.2 Format dels missatges

Tots els missatges del protocol DNS segueixen un esquema genèric format per cinc seccions:

- `Capçalera`
- `QUESTION SECTION`

  Conté la petició original que s'ha realitzat.
- `ANSWER SECTION`

  Conté els registres de la resposta a la petició realitzada.
- `AUTHORITY SECTION`

  Proporciona les dades dels servidors *autoritatius* per al domini sol·licitat.
- `ADDITIONAL SECTION`

  Informació addicional d'utilitat.

No totes les seccions estan presents en cada missatge. La capçalera indica quines seccions apareixen en el missatge.

Per tal d'interrogar un servidor DNS i observar-ne les respostes, es poden utilitzar utilitats com `nslookup` o `dig`.

2.3.3 Servidors mestres i esclaus

Per tal d'incrementar la fiabilitat del sistema DNS és necessari configurar dos o més servidors de noms per tal d'evitar que la caiguda d'un servidor no impliqui la impossibilitat de resoldre els *hosts* del domini que gestiona. Aquesta recomanació està documentada al [EBBP97] i és un requisit obligatori quan es tracta de servidors públics.

Per tant els servidors esclaus han de contenir les mateixes dades de zona que el servidor mestre (o primari), donat que en són igualment *autoritatius*. D'altra banda els servidors esclaus poden ser esclaus de només determinades zones i ser alhora primaris d'altres zones. Per tal de reduir la gestió que implica disposar de les mateixes dades actualitzades en primaris i esclaus, l'especificació de DNS permet realitzar transferència de zones entre servidors. D'aquesta manera el servidor primari conté la còpia mestra de les dades de zona i les distribueix de forma automàtica als esclaus a través del protocol de transferència de zones AXFR/IXFR.

La única diferència entre un servidor mestre i un esclau és la font d'on obté les dades. El servidor mestre obté les dades del sistema de fitxers —usualment local— mentre que el servidor esclau les obté a través de la transferència de zones.

Quan el ≪timer≫ *refresh* del SOA expira, el servidor esclau contacta amb el primari per comprovar si s'han produït canvis en la zona. En cas afirmatiu inicia la transferència de zona. Aquest procés però pot ser iniciat pel servidor primari si es degudament configurat per a que envii notificacions als esclaus quan es produeix un canvi en la zona.

La distinció entre mestres i esclaus no te cap implicació sobre la prioritat d'accés. Només defineix quin és el servidor que contindrà la copia mestra de les zones, la qual serà transferida a la resta.

2.3.4 Transferència de zones

Per tal de simplificar la gestió de diversos servidors que son *autoritatius* de les mateixes dades, l'especificació de DNS disposa d'un protocol per a realitzar la transferència de zones entre servidors mestres i esclaus. El temps en que aquesta transferència es realitza, és un factor determinant en la velocitat en que els canvis es propaguen a través d'Internet.

El sistema DNS disposa de dos mecanismes per a la transferència de zones. Aquests es veuen a continuació.

Transferència de zona completa (AXFR)

L'especificació de DNS, disposa que els servidors esclaus interroguen —en un interval de temps determinat pel valor *refresh* del registre SOA— al primari per tal d'esbrinar si s'hi han produït canvis (en les zones que l'esclau té configurades). Aquest valor s'estableix habitualment en 12 o més hores. És important que quan una zona mestra es modifiqui, el valor *serial number* s'actualitzi per tal que els esclaus determinin que s'ha produït una actualització. Els servidors esclaus determinaran en base a aquest paràmetre si cal que inicïin una petició de transferència de zona.

En la majoria dels casos les transferències de zona AXFR es realitzaran utilitzant el protocol TCP pel port 53, donat que les dades de zona superaran el límit establert de 512 octets per a l'us d'UDP.

Tal i com es veurà en el capítol 6, canviant el *backend* de BIND i utilitzant la replicació que ofereix OpenLDAP, la transferència de zones AXFR deixa de ser necessària. De la mateixa manera la recarrega de zones deixa de ser necessària.

Transferència de zona incremental (IXFR)

Per tal de reduir el temps i l'ample de banda i recursos requerits per a la transferència de zona, es va definir posteriorment a [Oht96] la transferència incremental. Aquest

mecanisme permet transferir només els registres que s'han vist modificats des de la última transferència. D'aquesta manera si només un o escassos registres s'han modificat, no farà falta transferir la zona sencera.

Per tal d'utilitzar aquest mecanisme cal que tant el primari com l'esclau suportin aquesta característica

Notificació (NOTIFY)

Per tal d'accelerar el procés de propagació dels canvis, el RFC 1996 defineix un mecanisme de notificació per part dels servidors primaris als esclaus quan una zona es carrega o actualitza. En rebre notificació, l'esclau comprovarà de nou si és necessària la petició d'una transferència de zona a través de la comprovació del *serial number*.

2.3.5 Actualització dinàmica

El procediment de modificació i recarrega de fitxers de zona es pot convertir en insostenible quan el servidor gestiona zones extenses i el volum de canvis assoleix nivells elevats. En aquestes circumstàncies el servidor requereix d'un temps important per tal de recarregar les zones, temps el qual deixa de respondre peticions. Per tal de resoldre aquesta problemàtica es presenten dues aproximacions arquitectòniques:

(a) Habilitar les actualitzacions en temps d'execució a través de l'actualització dels registres des d'una aplicació externa.

(b) Llegir directament les dades de zona —substituint així el *backend* en memòria— d'una base de dades que podrà ser actualitzada dinàmicament.

En el capítol 6 es descriu àmpliament aquesta aproximació, mentre que en l'apèndix B es descriu el desplegament d'aquesta solució utilitzant un directori i el protocol LDAP, junt als seus avantatges i mecanismes d'autenticació que s'utilitzen.

La solució adoptada pel RFC 2136 es basa en l'aproximació (a) i defineix el procés anomenat *Dinamic DNS*, on els registres poden ser actualitzats des d'una font externa. En aquest RFC es defineix el terme *Primary Master* per definir el servidor que apareix al registre SOA. Habitualment aquesta operació és descrita juntament amb algunes característiques de seguretat del DNS com els registres TSIG (*Transaction Signature*) i TKEY (*Transaction Key*). Quan es realitza l'actualització dinàmica, cap la possibilitat que una font maliciosa realitzi un enverinament dels fitxers de zona. Per evitar això els registres TSIG i TKEY s'utilitzen per tal d'autenticar la font de les peticions d'actualització assegurant la integritat de les dades. Tot i això el mecanisme TSIG presenta algunes deficiències:

– utilitza criptografía de clau compartida, el qual en dificulta el manteniment i escalabilitat;

– utilitza una funció de *hash* HMAC-MD5 de només 128 bits;

– no disposa de permisos d'accés, per tant qualsevol que conegui la clau pot modificar qualsevol registre.

Aquests problemes es solucionen utilitzant l'aproximació (b), on per exemple amb l'ús de LDAP es pot restringir l'accés a les modificacions i fer ús de certificats digitals per proveir d'un mecanisme segur d'autenticació.

2.3.6 Consideracions de seguretat

En l'operació dels servidors DNS cal tenir en compte algunes consideracions importants de seguretat. Aquestes afecten als procediments d'actualització de les dades d'un servidor. Com ja s'ha vist, els servidors esclaus utilitzen la transferència de zones definida a les especificacions de DNS, mentre que en l'actualització dinàmica (DDNS) aquesta es produeix per terceres parts.

En aquests processos d'actualització s'hi poden produïr amenaces de seguretat. Aquests es descriuen a continuació:

- **Transferència de zona**: El protocol de transferència de zona no estableix cap mecanisme d'autenticació més enllà de l'autenticació basada en *host* —això eś verificar l'adreça IP. Al no disposar alhora el protocol IPv4 d'autenticació, no es pot assegurar que en el procés d'actualització es produeixi una suplantació d'identitat del servidor mestre. En aquesta situació un servidor esclau podria sol·licitar una transferència de zona d'una font no legítima que podria resultar en l'enverinament de les dades *autoritatives* del servidor esclau.

  Per tant el mecanisme de transferència de zones no és segur. El sistema descrit en l'apèndix B presenta una de les solucions a aquest problema. L'ús dels registres TSIG i TKEY també ofereix un mètode de transmissió segura de les zones.

- **DDNS**: De forma anàloga, i com ja s'ha comentat, el procés d'actualització dinàmica cal que faci ús dels registres TSIG i TKEY per evitar la suplantació.

A banda de l'actualització, el el flux d'operació de DNS es presenten altres vulnerabilitats com:

- **Enverinament de *caché* remota**: L'enverinament de les dades de la *cache* d'un servidor —que es pot produïr a través de diversos mecanismes com la suplantació d'identitat— provoquen que les respostes a les dades *cachejades* siguin corruptes, e.g. les dades enverinades poden contenir informació falsificada amb objectius diversos com redirigir les peticions d'un nom de domini a una adreça IP il·legítima. DNSSEC presenta un mecanisme per evitar l'enverinament de la *caché*.

- **DNS *hijack***: El darrer problema en el flux d'operació de DNS es troba en el *resolver* del client i la comunicació amb el servidor DNS que aquest tingui configurat. Al no disposar tampoc d'autenticació, tècniques com la suplantació (ARP*spoofing* o IP *spoofing*) es poden emprar per segrestar les peticions i enviar informació falsificada a un client. DNSSEC soluciona aquest problema. El servidor BIND també permet la comunicació amb els clients sobre SSL/TLS, característica que evita el problema.

2.4 Modes especials d'operació

En aquesta secció es descriuen algunes configuracions avançades del sistema DNS que corresponen a diferents modes d'operació dels servidors. Ja s'ha vist en la secció 2.3.3 que un servidor DNS pot funcionar com a mestre o com a esclau d'una zona. Tanmateix també poden realitzar o no *cacheig* de les dades quan realitzen peticions recursives. Altres modes de funcionament es descriuen a continuació.

2.4.1 Servidor *forwarding (Proxy)*

En aquesta configuració un servidor reenvia totes les peticions a un altre servidor i en *cacheja* les respostes. Aquesta pràctica s'utilitza amb els següents proposits:

- En un servidor local, accelera les respostes i redueix el trànsit extern generat.

- En peticions recursives genera només una transacció. El servidor al qual reenvia la petició manejarà els *referrals* i s'encarregarà de realitzar la *recurssió*, retornant una única resposta.

Aquesta configuració genera una *caché* gran i redueix les consultes externes. També s'utilitza en servidors *stealth/split*, els quals es descriuen tot seguit.

2.4.2 Servidor *Stealth* (DMZ o *Split*)

Un servidor *stealth* es defineix com aquell que no apareix en cap registre NS públicament visible per un domini. Es tracta per tant d'un servidor intern. La figura 2.4 mostra un esquema de l'arquitectura d'aquesta configuració.

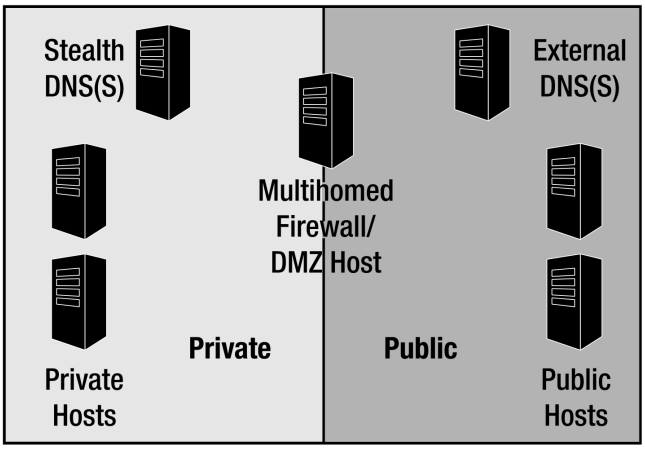

Figura 2.4: Arquitectura d'una configuració de servidor *stealth*.

Els servidors externs es configuren per proporcionar respostes autoritatives —per tant no responen a peticions recursives— i no realitzen *cacheig*. Les zones que aquests servidors tenen declarades es corresponen únicament a la part pública del domini (e.g. registres NS dels servidors públics, registres MX, registres A dels *hosts* de la DMZ,...).

D'aquesta manera el servidor intern (*stealth*) queda separat dels externs augmentant-ne la seguretat. Aquest comportament es pot assolir per exemple mitjançant les vistes que proporciona el servidor BIND. Tot i això, si el servidor de noms es veies compromès, les dades relatives a la part interna es veurien compromeses proporcionant informació de l'estructura interna de la xarxa. Per tant aquesta separació física n'augmenta la seguretat global.

El servidor intern donarà servei als usuaris i *hosts* interns incloent resolució de peticions recursives i *cacheig*.

2.4.3 Servidor *Authoritative-only*

Aquesta configuració s'utilitza per descriure dues característiques del servidor:

(i) El servidor respondrà amb respostes *autoritatives*.

(ii) El servidor no realitzarà *cacheig*

Un servidor s'utilitza habitualment en aquesta configuració en els casos següents:

- servidors públics en la configuració *stealth/split* per proporcionar seguretat perimetral;
- servidors d'alt rendiment, tals com els *root-servers*, servidors TLD o altres servidors amb un gran volum de peticions.

2.5 El servidor de noms BIND

El servidor BIND (*Berkeley Internet Name Domain*) és considerat l'estàndard *de facto* de les implementacions del sistema de noms de domini i és el servidor més utilitzat. Originàriament fou escrit per estudiants de la universitat de Berkeley, Califòrnia, però posteriorment va ser reescrit des de zero en la versió 9 per l'*Internet Systems Consortium* (ISC) per tal de redissenyar-ne l'arquitectura i afegir suport per DNSSEC i altres extensions de DNS.

BIND incorpora d'una llicència lliure de tipus permissiva: ISC license.

El servidor fa ús d'un *backend* en memòria que llegeix les dades dels fitxers de zona al carregar-se i les manté en memòria per tal de respondre les peticions que rep. Des de la versió 8 disposa però d'una API que permet modificar-ne el comportament per defecte i fer ús d'una base de dades. La versió 9.1 incorpora una API simplificada per a l'ús de bases de dades. El capítol 6 dóna més detalls sobre aquestes configuracions.

2.6 Seguretat

BIND ha estat fortament criticat pels problemes de seguretat que ha mostrat, especialment en versions més antigues. Fet pel qual es van iniciar altres projectes per a desenvolupar un servidor DNS més segur. Un exemple d'aquest és el servidor TinyDNS, escrit per l'autor de qmail, Daniel J. Bernstein.

La versió 9 de BIND, tot i estar reescrita des de zero presenta igualment greus problemes, la majoria dels quals corresponen a petits errors que cometria un programador novell. Una dura crítica de l'autor de TinyDNS on es detallen alguns dels problemes que ha presentat BIND es troba en [Ber]. El propi mantenidor del projecte, Paul Vixie, va caracteritzar el codi original com un programa desendreçat produït per una tropa de graduats beguts, de la Universitat de Berkeley.

2.6.1 BIND *chroot*

El fet de disposar de bind en un entorn *chroot*, *sandbox* o entorn de gàbia, garanteix que aquest no pugui llegir o escriure fora del directori base de la gàbia. Aquesta gàbia configura un entorn on l'arrel del sistema de fitxers es converteix en el directori base de la gàbia. D'aquesta manera es garanteix que un forat de seguretat en el servidor BIND no comprometi la totalitat de la màquina que l'executa.

La majoria de distribucions disposen del servidor BIND ja empaquetat en un entorn de gàbia.

2.7 Alternatives a BIND

Existeixen diverses alternatives al servidor BIND. Una de les quals a considerar es TinyDNS, el qual fou el segon servidor de noms de programari lliure més utilitzat el 2004. Es va oferir una recompensa de 1.000 dòlars a la primera persona que publiqués un informe verificable d'algun forat de seguretat en el servidor.

L'autor d'aquest servidor ha criticat fortament els *bugs* de seguretat que codi del servidor BIND ha contingut, fet pel qual va desenvolupar aquest servidor de DNS.

Capítol 3

DHCP. Protocol de configuració de hosts

El protocol DHCP (*Dinamic Host Configuration Protocol*) és un protocol de la capa d'aplicació de la pila de protocols TCP/IP, utilitzat per a l'assignació automàtica de configuracions de xarxa per als *hosts* connectats a una xarxa.

El protocol DHCP està originalment especificat en l'RFC 1531 i actualitzat en l'RFC 1541 [Dro93] com una extensió del protocol BOOTP i afegeix la capacitat d'assignació d'adreces de xarxa reutilitzables així com altres configuracions.

Es tracta d'un protocol de tipus client–servidor on el servidor DHCP designat assigna adreces i paràmetres de configuració al *hosts* clients, que es s'auto-configuren de forma dinàmica. Un cop configurats els clients podran comunicar-se amb la resta de *hosts* de la xarxa.

El protocol DHCP utilitza UDP per al transport i els ports 67 (servidor) i 68 (client).

El format dels missatges del protocol, està basat en els missatges de BOOTP i per tant es permet l'interoperabilitat entre els clients BOOTP i els servidors DHCP. Amb l'ús dels *relay agents* de BOOTP, s'evita haver de disposar d'un servidor DHCP en cada segment de xarxa (*o VLAN*).

Un agent de *relay* és un *host* o *router* que reenvia els missatges DHCP entre clients i servidors. Aguests agents de *relay* s'encarregaran de reenviar els missatges DHCP entre diferents xarxes.

El protocol disposa de tres mecanismes per a l'assignació de les adreces: *assignació automàtica*, *assignació dinàmica* i *assignació manual*:

- *Assignació automàtica*: El servidor assigna una adreça a un host de forma permanent.

- *Assignació manual*: Una adreça és assignada al *host* client segons una configuració preestablerta per l'administrador.
- *Assignació dinàmica*: En aquest cas el servidor assigna una adreça per un període limitat de temps.

Segons la política d'administració, s'utilitzarà un o altre mecanisme o una combinació dels mateixos.

El mecanisme d'assignació dinàmica és l'únic que permet la reutilització de les adreces que ja no són requerides pels *hosts* als quals han estat assignades.

3.1 Assignació dinàmica

El mecanisme d'assignació garanteix que l'adreça no s'assignarà de nou durant el període establert i que intentarà retornar la mateixa adreça cada cop que un client envia una petició. El temps en que una adreça és assignada a un client s'anomena *lease*.

El client pot estendre aquest *lease* amb peticions subsegüents. També pot enviar un missatge de *release* per tal d'alliberar el *lease* retornant aquesta adreça al *pool* d'adreces disponibles o inclús sol·licitar una assignació permanent.

Un cop el *lease* ha expirat, l'adreça retorna al *pool* d'adreces disponibles i per tant pot ser reutilitzada. Abans de reassignar una adreça, el servidor hauria de comprovar que l'adreça no estigui en ús, per exemple a través de l'enviament d'un paquet ≪ICMP echo request≫ o del protocol ARP. De la mateixa manera, el client n'hauria de comprovar també l'ús —a través del protocol ARP per exemple.

3.2 Objectius en el disseny

Els següents objectius van ser definits en el disseny del protocol:

- El DHCP és un mecanisme i no una política. Ha de permetre als administradors controlar les configuracions.
- Un client ha de poder descobrir els paràmetres de configuració de forma automàtica, sense intervenció de l'usuari.
- No hauria de requerir un servidor per a cada *subnet*. Cal que el DHCP funcioni a través de la xarxa mitjançant els agents de *relay* de BOOTP.

- Un client haurà d'estar preparat per rebre diverses respostes a una petició de configuració. En determinats entorns diversos servidors DHCP es sobreposen per tal d'augmentar-ne la fiabilitat.
- El DHCP ha de coexistir amb els *hosts* configurats de forma estàtica i les implementacions dels protocols de xarxa existents.
- Ha de proporcionar servei als clients BOOTP existents.
- Cal que garanteixi que una adreça de xarxa no estigui en ús per més d'un client DHCP de forma simultània.
- Mantenir la configuració d'un client després que aquest es reinicii.
- Permetre assignació d'adreces i configuració a nous clients.
- Suportar l'assignació permanent de paràmetres a determinats clients específics.

3.3 Flux d'interacció client–servidor

En el procés d'assignació de les adreces, client i servidor s'envien una sèrie de missatges. El flux d'interacció pot variar en funció de l'estat previ del client. En la taula 3.1 es descriuen aquests missatges. A continuació se'n descriu el flux d'interacció:

3.3.1 Reserva d'una adreça nova

1. Un client envia un missatge DHCPDISCOVER amb adreça destí *broadcast* a la xarxa física. En cas que es disposi d'agents de *relay*, aquestos retransmeten la petició a les altres xarxes. L'adreça IP origen del client s'estableix al valor especial 0.0.0.0.
2. Cada servidor que rep la petició respon amb un missatge DHCPOFFER que inclou l'adreça oferta i altres paràmetres de configuració.
3. El client rep un o diversos missatges DHCPOFFER amb la oferta de cadascun dels servidors i n'escull un servidor basant-se en els paràmetres que se li ofereixen. El client envia un missatge DHCPREQUEST amb adreça destí *broadcast*, que inclou l'identificador del servidor del qual ha acceptat la configuració, i que pot incloure altres opcions especificant els paràmetres de configuració desitjats (cal que l'adreça sigui la que s'ha ofertat). L'adreça IP origen del client s'estableix al valor especial 0.0.0.0. El missatge és reenviat pels agents de *relay* en cas que se'n disposi d'algun. En el cas que es produeixi un *timeout* en el client en l'espera d'un missatge DHCPOFFER, aquest torna a iniciar el procés enviant un missatge DHCPDISCOVER.

Taula 3.1: Missatges DHCP

Missatge	**Ús**
DHCPDISCOVER	El client envia una petició amb destí adreça de *broadcast* per tal de localitzar els servidors disponibles.
DHCPOFFER	Els servidors responen la petició del client amb els paràmetres de configuració oferts.
DHCPREQUEST	Un client envia aquest missatge als servidors amb una de les finalitats següents: (a) sol·licitar els paràmetres oferts per un dels servidors, descartant les ofertes de la resta; (b) confirmar la validesa d'una adreça ja assignada prèviament; (c) estenent el *lease* sobre una adreça en particular.
DHCPACK	El servidor envia al client les configuracions de xarxa, inclosa l'adreça prèviament ofertada.
DHCPNAK	El servidor envia aquest missatge al client per indicar que la noció de xarxa que aquest té és incorrecta (e.g. el client s'ha mogut a una altra *subnet* o VLAN) o bé que l'adreça que sol·licita ja està en ús.
DHCPDECLINE	Un client indica al servidor que l'adreça ja està en ús.
DHCPRELEASE	Un client indica al servidor que allibera l'adreça que té assignada, destruint-ne el *lease* i deixant-la lliure.
DHCPINFORM	Un client sol·licita només els paràmetres de configuració local; el client ja disposa d'una adreça de xarxa configurada.

4. Els servidors reben el missatge DHCPREQUEST del client. Aquells que no han estat seleccionat utilitzen aquest missatge com a notificació. El servidor seleccionat respon amb un missatge DHCPACK que conté els paràmetres de configuració. En cas que per algun motiu no es pugui satisfer la sol·licitud (e.g. l'adreça ja està en ús per un altre *host*), el servidor envia un missatge DHCPNAK.

5. El client rep el missatge DHCPACK amb els paràmetres de configuració dels quals n'*hauria* de realitzar una comprovació final (e.g. enviar una sol·licitud ARP per comprovar que l'adreça no està en ús) i n'anota el temps de *lease*. En cas que detecti que l'adreça ja està en ús envia un missatge DHCPDECLINE al servidor i reinicia el procés de configuració. El client *hauria* d'esperar com a mínim deu segons abans de reiniciar el procés de configuració per evitar generar un excés de trànsit en cas de bucles.

   En cas que el client hagi rebut un missatge DHCPNAK reinicia el procés de

configuració. En cas que es produeixi un *timeout* en l'espera d'una resposta DHCPACK/DHCPNAK el client retransmet el missatge DHCPREQUEST d'acord a un algorisme de retransmissió que n'estima el temps.

6. El client pot renunciar al seu *lease* a través de l'enviament d'un missatge DHCPRELEASE al servidor.

3.3.2 Reutilització d'una adreça previament assignada

En aquest cas el client és qui sol·licita l'adreça en particular que prèviament li ha estat assignada i de la qual en conserva el *lease*. Alguns dels passos descrits anteriorment són omesos en aquest cas.

Concretament el client s'inicia en el pas (3) enviant un missatge DHCPREQUEST amb l'adreça prèviament assignada.

3.4 Paràmetres del client

En la taula 3.2 es llisten els paràmetres de configuració que un servidor DHCP gestiona. Cal assenyalar però que no tots els clients faran ús de tots aquests paràmetres. Per tal de reduïr el nombre de paràmetres que el servidor ha d'enviar s'utilitzen dos estragègies:

(a) La major part dels paràmetres tenen valors predefinits. Si un client no rep el valor de configuració d'aquests paràmetres utilitza els valors per defecte.

(b) En el missatge inicial DHCPDISCOVER/DHCPREQUEST el client pot especificar quins són els valors de configuració que desitja rebre.

Nota:

El protocol DHCP no està dissenyat per a la configuració de routers.

Taula 3.2: Paràmetres de configuració de *host*

Paràmetres de la capa IP (per host):	
Be a router	on/off
Non-local source routing	on/off
Policy filters for non-local source routing	(list)
Maximum reassembly size	integer
Default TTL	integer
PMTU aging timeout	integer
MTU plateau table	(list)
Paràmetres de la capa IP (per interfície):	
IP address	(address)
Subnet mask	(address mask)
MTU	integer
All-subnets-MTU	on/off
Broadcast address flavor	0x00000000/0xffffffff
Perform mask discovery	on/off
Be a mask supplier	on/off
Perform router discovery	on/off
Router solicitation address	(address)
Rutes per defecte (llistat):	
router address	(address)
preference level	integer
Rutes estàtiques (llistat):	
destination	(host/subnet/net)
destination mask	(address mask)
type-of-service	integer
first-hop router	(address)
ignore redirects	on/off
PMTU	integer
perform PMTU discovery	on/off
Paràmetres de la capa d'enllaç de dades (per interfície):	
Trailers	on/off
ARP cache timeout	integer
Ethernet encapsulation	(RFC 894/RFC 1042)
Paràmetres TCP (per *host*):	
TTL	integer
Keep-alive interval	integer
Keep-alive data size	0/1

Capítol 4

Estructura de xarxa de la UdL

En aquest capítol es dóna una visió general de les característiques de la xarxa de comunicacions de la Universitat de Lleida (UdL). Se'n descriu l'arquitectura de xarxa, tot centrant-se en les parts on resideixen els servidors de noms i es descriuen els procediments de gestió per al manteniment tant dels servidors de noms com del DHCP. Aquesta gestió es contrasta amb els procediments requerits mitjançant l'ús del sistema Sauron i se n'avaluen els beneficis.

4.1 Estructura de xarxa

La xarxa de la UdL està formada per cinc campus ubicats en diferents punts de Lleida:

- Campus Cappont
- Campus Rectorat
- Campus Ciències de la Salut / UDHAV (Hospital Arnau de Vilanova)
- Campus ETSEA
- Campus INEFC

Aquests cinc campus estan units conformant una topologia d'estrella amb centre al node de Rectorat, on s'hi troba també el centre de processament de dades (CPD) de la universitat i i l'àrea de comunicacions i sistemes.

La UdL disposa de cinc classes C d'adreçament públic assignades per RedIRIS[1]:

- 193.144.8.0/24

[1] RedIRIS, és la xarxa nacional de recerca i educació espanyola.

- 193.144.9.0/24
- 193.144.10.0/24
- 193.144.11.0/24
- 193.144.12.0/24

Inicialment cadascuna de les classes C anteriors es corresponia a un campus, però amb la implantació de NAT[2] l'any '96, aquesta organització inicial s'ha vist modificada.

Es disposa també dels dominis públics,

- udl.cat,
- udl.es,

i s'utilitza com a domini intern udl.net, tot i que aquest no pertany a la UdL i només s'utilitza internament.

Pel que fa a l'adreçament privat s'utilitzen les xarxes 10.0.0.0/8 i 172.16.0.0/16 per a les VLANs ≪Intranet≫ i ≪Aules≫ respectivament. S'utilitzen també altres xarxes privades amb adreçament de classe C com la VLAN de VoIP o algunes VLANs que no s'encaminen, com la VLAN de gestió dels equips de xarxa. Les darreres però s'exclouen dels diagrames d'aquest capítol donat que són xarxes que no s'encaminen i per tant romanen aïllades i són irrellevants per al desenvolupament d'aquest treball, que es centra en els serveis de DNS i DHCP de les VLANs ≪Intranet≫ i ≪Aules≫ i el DNS públic.

Tal i com s'ha comentat en el capítol 2, són necessaris al menys dos servidors de noms per incrementar la fiabilitat i proporcionar tolerància a fallades. La universitat disposa de diversos servidors de noms, que alhora serveixen per proporcionar un mecanisme de balanceig de càrrega.

Es disposa de tres servidors de noms per a la resolució interna, aquests són:

- DNS1 (dns1.udl.net),
- DNS2 (dns2.udl.net),
- DNS3 (dns3.udl.net);

[2]NAT (Network Address Translation) és un mecanisme per tal de modificar (traduir) les adreces de les capçaleres dels datagrames IP, en aquest cas per traduir les adreces públiques a les corresponents adreces privades per als servidors.

i de dos servidors de noms externs:

- Gardeny (gardeny.udl.cat),
- Marraco (marraco.udl.cat).

Els servidors DNS1, DNS2 i DNS3 són utilitzats per a la resolució de noms de la VLAN ≪Intranet≫, mentre que DNS3 és utilitzat també per a la VLAN ≪Aules≫ pel mateix propòsit. Tots tres servidors són *autoritatius* per la zona udl.net i per les zones udl.cat i udl.es. D'aquesta manera la resolució de noms per als *hosts* de la universitat (tant amb adreçament públic com privat) es realitza pels servidors de noms propis. Aquests servidors hauran de permetre també *recursion* per a les xarxes internes (intranet i aules), es a dir la realització de peticions recursives per a la resolució de noms de zones no autoritatives, per tal de permetre als *hosts* d'aquestes xarxes resoldre qualsevol nom. Per tal de no sobrecarregar els servidors de noms, la resolució de peticions recursives es permet només per a les xarxes de la UdL.

Els servidors DNS1 i DNS3 corren alhora el servei de DHCP per a les VLANs ≪Intranet≫ i ≪Aules≫ respectivament.

Amb la implantació del sistema Sauron, s'aprofitarà per unificar els dos servidors de DHCP que actualment serveixen de forma exclusiva en cadascuna de les VLANs, per un únic servidor. Prenent profit que DNS3 serveix a les dues VLANs, i per tant té una interfície de xarxa a cada VLAN, el DHCP s'ubicarà en aquesta màquina.

La figura 4.1 mostra un esquema lògic de la xarxa descrita on s'hi mostra també la disposició dels servidors.

La configuració actual del sistema de noms es basa en un servidor primari (o mestre) i dos servidors secundaris (o esclaus). El servidor primari (DNS1) conté les dades de zona i notifica dels canvis a través de la transferència de zones als esclaus (DNS2 i DNS3). Tal i com es descriu en el capítol 8 i l'apèndix B, amb la implantació de Sauron, l'ús d'un directori com a *backend* de dades del DNS i la replicació que proporciona LDAP, tots tres servidors passaran a ser primaris, eliminant per tant la transferència de zones entre servidors. Un diagrama d'aquesta arquitectura es mostra en la figura B.1 de l'apèndix B. Alguns dels avantatges d'aquesta configuració es descriuen en el capítol 6.

Cal esmentar que a les VLANs ≪Intranet≫ i ≪Aules≫ no es realitza *subneting* i el domini de *broadcast* es correspon per tant a 10.0.0.0/8 i 172.16.0.0/16 respectivament. Així doncs, la que es mostra en les taules 4.1 i 4.2 és només una forma d'organització de les adreces en departaments. Cal insistir en que les màscares de xarxa són /8 per intranet i /16 per aules, i que les especificades en les taules es defineixen tan sols per a l'ús intern de Sauron; a partir de les quals permetrà la gestió d'adreces en funció del departaments i realitzar accions com l'assignació automàtica d'adreces en funció a departament o el moviment de *hosts* entre departaments entre altres.

L'encaminament de la xarxa és força senzill i està configurat de forma estàtica:

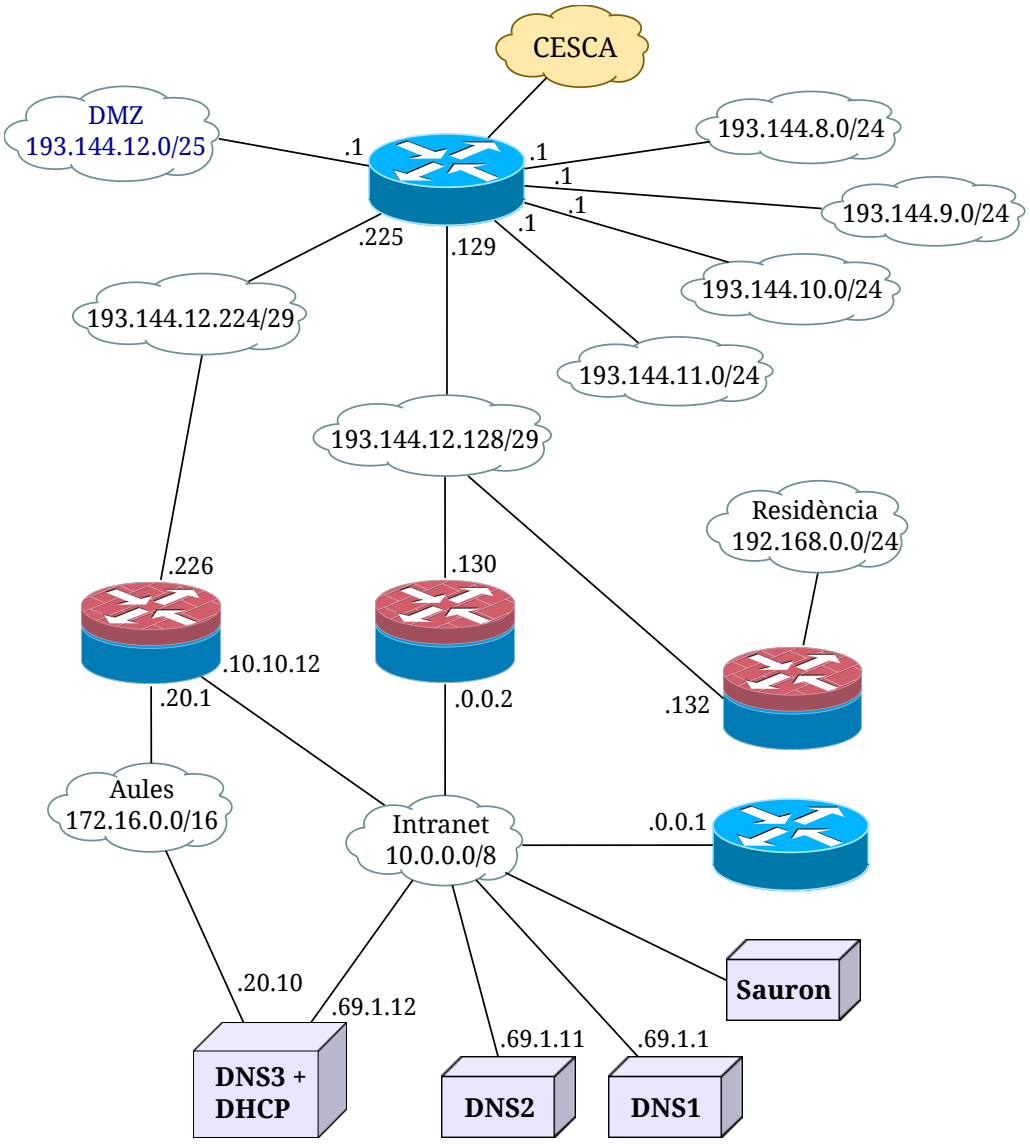

Figura 4.1: Mapa lògic de xarxa de la UdL

– Per una banda el *router* de la intranet conté en la seva taula d'encaminament una entrada per la xarxa d'aules, i com a ruta per defecte el *firewall* intern (10.0.0.2). Aquest firewall (10.0.0.2) encamina tot el trànsit al *router* extern (193.144.12.129).

– El *router* extern encamina tot el trànsit destinat a les xarxes externes a la UdL cap al proveïdor, en aquest cas el CESCA[3].

[3]El CESCA (Centre de Serveis Científics i Acadèmics de Catalunya) proveeix d'una xarxa que con-

4.2 Gestió de la infraestructura

Actualment el DNS intern conté vora 10.000 entrades de *hosts* hi més de 3.000 *hosts* estan registrats al DHCP amb una adreça IP fixada. Aquest elevat nombre d'equips provoca que la correcta gestió dels mateixos esdevingui un aspecte important en la gestió de la infraestructura tecnològica.

Cada modificació, moviment o adició d'un nou equip d'usuari —això inclou tant equips de PDI, PAS com equips d'aules— cal que es vegi reflectida en els servidors de noms i DHCP.

Actualment, en el cas de la UdL, aquesta gestió es realitza en dos nivells:

1. El personal de suport informàtic de cada campus (operadors de campus), que són els encarregats de la gestió dels equips d'usuari, sol·liciten a través del sistema d'assistències que disposa la universitat les modificacions necessàries.

   Quan es tracta d'afegir nous hosts, es disposa d'una petita aplicació Web que ajuda en la selecció de les adreces IP corresponents. Tot i això, es tracta només d'un formulari per obtenir la següent adreça disponible (segons departament) i enviar les dades en forma d'assistència a l'àrea de sistemes i comunicacions.

2. El personal de sistemes i comunicacions rep les assistències introduïdes pels operadors de campus i procedeix a realitzar les modificacions de forma manual sobre els servidors de noms i DHCP i recarregar-ne les configuracions.

La implantació del sistema Sauron a la UdL *elimina aquest procés de dos nivells*, delegant de forma completa i controlada la gestió dels equips des de la vessant del DNS/DHCP als propis operadors de campus.

4.2.1 Gestió dels serveis

Com ja s'ha comentat la gestió dels serveis de DNS i DHCP es realitza de forma manual. Aquesta aproximació és útil per a la gestió d'infraestructures reduïdes i quan un únic administrador és qui s'encarrega de la gestió.

Un dels problemes que presenta en un entorn com el de la UdL aquesta gestió manual, és el de la modificació concurrent de dades per part de diversos administradors. Altres problemes que suposa, degut a l'elevat nombre de *hosts*, és la consistència de les configuracions, que al llarg del temps poden resultar en dades obsoletes, inconsistències entre registres,....

A banda d'això la coherència en les configuracions és important per tal d'evitar futurs problemes, fet que es pot incomplir quan diverses persones en modifiquen les dades.

necta totes les universitats catalanes i que forma part de RedIRIS.

Per tant l'ús d'un sistema com Sauron, soluciona els problemes citats i proporciona una manera senzilla d'operar amb les dades dels serveis. El desplegament del sistema el detalla en el capítol 8.

Taula 4.1: Distribució de les IPs en ús de la VLAN «Aules»

Rang de xarxa	Inici – Fi	Núm. adreces	Facultat
DHCP			
172.16.100.0/22	.100.0 → .103.255	1024 @	Rang dinàmic DHCP
Aules/Biblioteques			
172.16.0.0/22	.0.0 → .3.255	1024 @	EPS-AULARI
172.16.4.0/22	.4.0 → .7.255	1024 @	ETSEA
172.16.8.0/23	.8.0 → .9.255	512 @	FDE
172.16.10.0/23	.10.0 → .11.255	512 @	LLETRES
172.16.12.0/22	.12.0 → 15.255	1024 @	FCE
172.16.16.0/22	.16.0 → .19.255	1024 @	CSALUT-UDHAV
172.16.20.0/22	.20.0 → 23.255	1024 @	SERVEIS-ASIC
172.16.24.0/24	.24.0 → .24.255	256 @	LAB-LINGUISTICA
172.16.25.0/24	.25.0 → 25.255	256 @	FDE-ARANZADI
172.16.30.0/24	.30.0 → .30.255	256 @	AUDIOVISUALS
172.16.31.0/24	.31.0 → .31.255	256 @	ICE
172.16.40.0/24	.40.0 → .40.255	256 @	BIB-IMPRE
172.16.41.0/24	.41.0 → 41.255	256 @	BIB-RECTORAT
172.16.42.0/24	.42.0 → .42.255	256 @	BIB-CAPPONT
172.16.43.0/24	.43.0 → .43.255	256 @	BIB-ETSEA
172.16.44.0/24	.44.0 → .44.255	256 @	BIB-CSALUT
172.16.45.0/24	.45.0 → .45.255	256 @	BIB-FCE
172.16.50.0/24	.50.0 → .50.255	256 @	PL-RECTORAT
172.16.51.0/24	.51.0 → .51.255	256 @	PL-CAPPONT
172.16.52.0/24	.52.0 → .52.255	256 @	PL-ETSEA
172.16.53.0/24	.53.0 → .53.255	256 @	PL-CSALUT
172.16.54.0/24	.54.0 → .54.255	256 @	PL-FCE

Taula 4.2: Distribució de les IPs en ús de la VLAN «Intranet»

Rang de xarxa	Inici – Fi	Núm. adreces	Departament
Equipament			
10.0.0.0/24	.0 → .255	256 @	Equips networking/ firewalling
10.10.10.0/24	.0 → .255	256 @	
Servidors			
10.50.90.0/24	.0 → .255	256 @	Tunels
10.50.91.0/24	.0 → .255	256 @	TunelsW
10.50.92.0/24	.0 → .255	256 @	TunelVPN
10.59.1.0/24	.0 → .255	256 @	
10.66.66.0/24	.0 → .255	256 @	DRP, DR1
10.69.0.0/16	.0.0 → .255.255	65536 @	Servidors
10.70.0.0/24	.0 → .255	256 @	Servidors LDAP
10.95.0.0/24	.0 → .255	256 @	W95
DHCP			
10.100.0.0/20	.0.0 → .15.255	4096 @	Rang dinàmic DHCP
Departaments			
10.50.1.0/24	.0 → .255	256 @	ASIC
10.50.2.0/24	.0 → .255	256 @	ARH

Continua a la pàgina següent

Taula4.2 – continuació de la pàgina anterior

Rang de xarxa	Inici – Fi	Núm. adreces	Departament
10.50.3.0/24	.0 → .255	256 @	AGA
10.50.4.0/24	.0 → .255	256 @	CEST
10.50.5.0/24	.0 → .255	256 @	DPUB
10.50.6.0/24	.0 → .255	256 @	AEGERN
10.50.7.0/24	.0 → .255	256 @	INEFC
10.50.8.0/24	.0 → .255	256 @	SUICRI
10.50.9.0/24	.0 → .255	256 @	AFIN
10.50.10.0/24	.0 → .255	256 @	ICE
10.50.11.0/24	.0 → .255	256 @	MAT
10.50.12.0/24	.0 → .255	256 @	UGERHEUP
10.50.13.0/24	.0 → .255	256 @	QUIMICA
10.50.14.0/24	.0 → .255	256 @	HBJ
10.50.15.0/24	.0 → .255	256 @	MACS
10.50.16.0/24	.0 → .255	256 @	IRBLLEIDA
10.50.17.0/24	.0 → .255	256 @	CMB
10.50.18.0/24	.0 → .255	256 @	PVCF
10.50.19.0/24	.0 → .255	256 @	APAT
10.50.20.0/24	.0 → .255	256 @	EAGROF
10.50.21.0/24	.0 → .255	256 @	PRODAN
10.50.22.0/24	.0 → .255	256 @	DAL
10.50.23.0/24	.0 → .255	256 @	CGX
10.50.24.0/24	.0 → .255	256 @	DPRIV
10.50.25.0/24	.0 → .255	256 @	UETSEA
10.50.26.0/24	.0 → .255	256 @	GEOSOC
10.50.27.0/24	.0 → .255	256 @	RECTOR
10.50.28.0/24	.0 → .255	256 @	FILCAT
10.50.29.0/24	.0 → .255	256 @	GERENCIA
10.50.30.0/24	.0 → .255	256 @	BIBLIO
10.50.31.0/24	.0 → .255	256 @	SCP
10.50.32.0/24	.0 → .255	256 @	FDE
10.50.33.0/24	.0 → .255	256 @	FLL
10.50.34.0/24	.0 → .255	256 @	TECAL
10.50.350/24	.0 → .255	256 @	HIST
10.50.36.0/24	.0 → .255	256 @	LABOR
10.50.37.0/24	.0 → .255	256 @	FICUS
10.50.38.0/24	.0 → .255	256 @	ESP
10.50.39.0/24	.0 → .255	256 @	INT
10.50.40.0/24	.0 → .255	256 @	FILCEF
10.50.41.0/24	.0 → .255	256 @	SG
10.50.42.0/24	.0 → .255	256 @	PIP
10.50.43.0/24	.0 → .255	256 @	DIDESP
10.50.44.0/24	.0 → .255	256 @	ACCSRV
10.50.45.0/24	.0 → .255	256 @	UGERHCE
10.50.46.0/24	.0 → .255	256 @	IVIDEO
10.50.47.0/24	.0 → .255	256 @	SPUB
10.50.48.0/24	.0 → .255	256 @	UDHAV
10.50.49.0/24	.0 → .255	256 @	MULT
10.50.50.0/24	.0 → .255	256 @	ECONAP
10.50.51.0/24	.0 → .255	256 @	
10.50.52.0/24	.0 → .255	256 @	BIBREC

Continua a la pàgina següent

Taula4.2 – continuació de la pàgina anterior

Rang de xarxa	Inici – Fi	Núm. adreces	Departament
10.50.53.0/24	.0 → .255	256 @	BIBETS
10.50.54.0/24	.0 → .255	256 @	DIEI
10.50.55.0/24	.0 → .255	256 @	SEU
10.50.56.0/24	.0 → .255	256 @	BIBCCS
10.50.57.0/24	.0 → .255	256 @	HAHS
10.50.58.0/24	.0 → .255	256 @	UGERHCS
10.50.59.0/24	.0 → .255	256 @	BIBFCE
10.50.60.0/24	.0 → .255	256 @	CULTURALS
10.50.61.0/24	.0 → .255	256 @	UGERHLL
10.50.62.0/24	.0 → .255	256 @	MEDICINA
10.50.63.0/24	.0 → .255	256 @	CIRURGIA
10.50.64.0/24	.0 → .255	256 @	UGERHDE
10.50.65.0/24	.0 → .255	256 @	CS
10.50.66.0/24	.0 → .255	256 @	SALUT
10.50.67.0/24	.0 → .255	256 @	SDG
10.50.68.0/24	.0 → .255	256 @	SLT
10.50.69.0/24	.0 → .255	256 @	OFICINARDI
10.50.70.0/24	.0 → .255	256 @	
10.50.71.0/24	.0 → .255	256 @	INF
10.50.72.0/24	.0 → .255	256 @	BIBCAP
10.50.73.0/24	.0 → .255	256 @	MEX

Capítol 5

Sauron. Gestió de hosts i servidors de noms

En aquest capítol es presenta Sauron, un sistema de gestió de servidors de noms i DHCP de programari lliure. Se'n donarà una descripció de les característiques i arquitectura i se'n farà un anàlisi seguint alguns models d'ànalisi de projectes FLOSS. Finalment s'introdueixen les correccions i millores que s'han implementat sobre el sistema.

5.1 Descripció general

Sauron és un sistema de programari lliure que facilita la gestió dels *hosts* d'una organització de grans dimensions —com és la pròpia UdL— d'una forma senzilla i segura a través d'una interfície Web. Es tracta d'un sistema escalable per a la gestió dels servidors de noms, DHCP i impressores. El sistema genera de forma dinàmica els fitxers de zona i configuracions per als servidors ISC BIND i ISC DHCP a partir de les dades que emmagatzema en una base de dades central.

Sauron es va dissenyar i desenvolupar al Centre de Computació de la Universitat de Jyvaskyla, Finlàndia, per Timo Kokkonen entre 1999 i 2004. Tot i que en seu origen pretenia proporcionar únicament una interfície Web per a la gestió dels fitxers de zona, va esdevenir en un sistema complet de gestió de servidors DNS i DHCP.

El sistema segueix actualment en desenvolupament, tot i que des que el seu autor T. Kokkonen deixà el seu lloc al centre de computació el 2004[1], el desenvolupament sembla avançar a un ritme lent. El 2010 es va alliberar la versió 0.7.3 —que incorpora un conjunt de canvis i característiques menors i la correcció de diversos problemes—

[1] Informació extreta de l'arxiu de les llistes de correu de Sauron

cinc anys després de l'anterior versió. La branca 0.7 correspon a la branca de desenvolupament, però els usuaris han reportat que funciona millor que la branca estable (0.6), la darrera versió de la qual (0.6.2) es va alliberar el 2003, poc després que el codi de Sauron s'alliberés públicament.

Sauron es va publicar el gener de 2003 sota els termes de la *GNU General Public License* (GPLv2), en la versió 0.6.0.

Com a referència, Sauron s'utilitza en producció en la xarxa de la Universitat de Jyvaskyla, gestionant un campus amb més de 10.000 *hosts*. Els propis administradors d'aquesta xarxa comenten que no els seria possible que els aproximadament 160 empleats de suport informàtic repartits entre facultats, residències, etc. poguessin tenir accés i gestionar directament d'una forma senzilla i controlada les dades del DNS i DHCP sense una eina com Sauron.

Aquest va ser un dels punts clau en els principis de disseny de Sauron: permetre la delegació de drets d'administració basats en usuaris i grups.

Així Sauron permet manejar de forma senzilla i concurrent a través d'una interfície d'usuari Web, les dades relatives als hosts d'una institució. L'ús de Sauron ajuda a una gestió més eficaç i segura de les dades dels servidors de noms i DHCP ja que evita haver de treballar amb extensos fitxers de zona que a mesura que creixen i són mantinguts per diferents administradors, tendeixen a «enrebessar-se» induint a errors o males configuracions que sovint són passades per alt.

Tal i com es descriu en 4, la implantació de Sauron a la UdL permetrà simplificar tot el procés actual de gestió.

Existeixen també eines comercials com Alcatel-Lucent VitalQIP, però que suposen una despesa important degut al cost de llicència; que creix en funció del nombre d'objectes que hagi de gestionar, i per tant, de les dimensions de la infraestructura tecnològica que gestionen.

5.1.1 Característiques

Algunes de les característiques rellevants de Sauron es llisten a continuació:

- Interfície Web amb control d'accés basat en usuaris i grups.
- Interfície de línia de comandes per tasques d'administració d'alt nivell.
- *Log* d'activitat del sistema.
- Generació dinàmica d'*alias* (CNAME).
- Permet generar de forma automàtica les zones inverses.
- Suport per a la delegació de *subnets* en zones inverses (veieu 2.2.5 per a més detalls).

- Permet definir data d'expiració per usuaris, *hosts* i *alias*.
- Suport per al protocol DHCP *failover*.
- Plantilles per a la creació ràpida de determinats registres (e.g. MX).
- Verificació automàtica de les configuracions generades (mitjançant les utilitats externes de BIND/DHCP).
- Exportació de *hosts* en format CSV.

En quant a les funcionalitats per a la gestió, permet entre d'altres operacions, copiar zones senceres i copiar/moure *hosts* entre diferents *subnets* a través de la mateixa interfície Web.

Algunes de les funcionalitats que ofereix la interfície de línia de comandes són:

- Gestió d'usuaris i grups
  - bloqueig temporal d'usuaris/grups,
  - restriccions basades en *subnets*,
  - restriccions basades en nom de host,
  - restriccions basades en rang IP,
  - restriccions en base a nivell de privilegis associat,
  - restriccions en base a grup de *host*,
  - privilegis basats en *flag*,
  - ...
- Importació global de les dades de DNS i DHCP.
- Importació de hosts a través de fitxers de zona i importació de dades de topologia de DHCP.
- Exportació de hosts per a una o totes les zones d'un *server* en format CSV, exportació de llistats d'IPs en ús per a una o diverses *subnets*,....
- Modificació de hosts de forma massiva (moure, reanomenar, esborrar,...).
- Actualització de hosts des de les dades d'un fitxer CSV.
- ...

5.2 Arquitectura

El sistema Sauron està escrit en Perl pràcticament en la seva totalitat i utilitza alguns paquets addicionals del repositori CPAN[2]. Un llistat d'aquests paquets addicionals es troba en la taula A.2 de l'apèndix A. L'arquitectura de Sauron es compon de forma general dels següents components (la figura 5.1 en mostra un diagrama):

[2]CPAN (*Comprehensive Perl Archive Network*), és un repositori de mòduls i documentació per a Perl. http://www.cpan.org

- el *backend* del sistema, que gestiona tots els processos entorn al mateix;
- una interfície de línia de comandes (CLI) per a tasques d'administració del sistema/gestió massiva de *hosts*,...;
- una base de dades relacional (PostgreSQL) per a l'emmagatzemament de dades tant del propi sistema com dels servidors que gestiona;
- un mòdul per a la generació de les configuracions dels serveis gestionats, que fa ús del backend i obté la informació de la BD;
- una interfície Web que permet gestionar la la informació dels servidors i que es correspon a la interfície d'usuari del sistema.

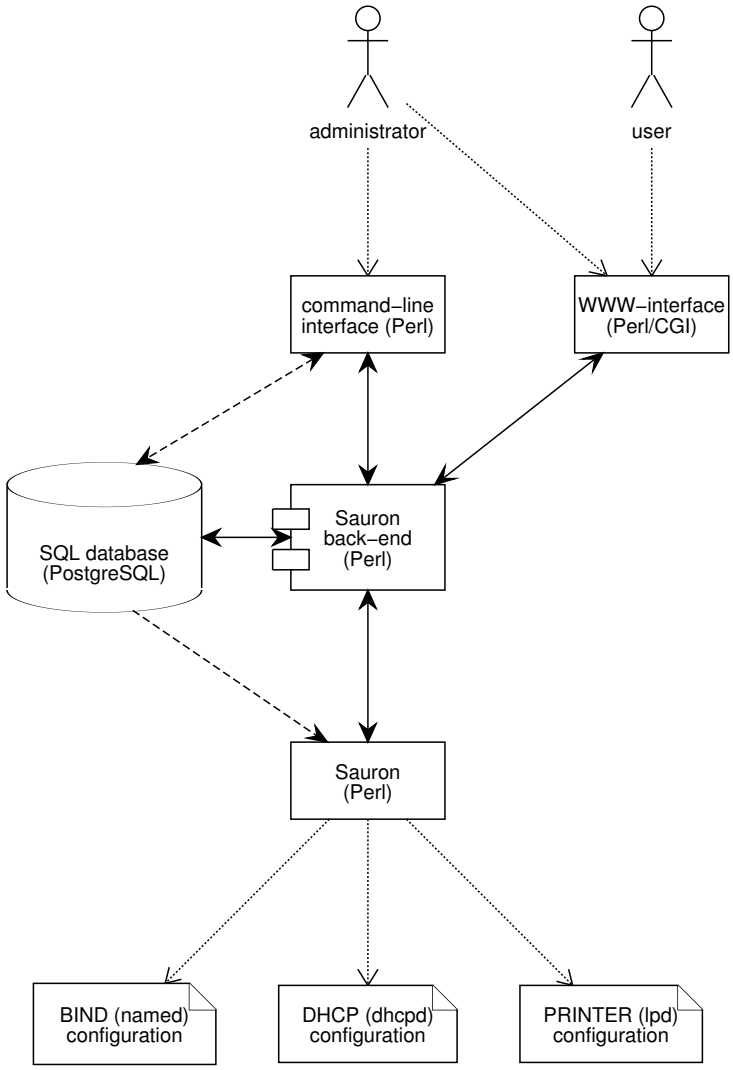

Figura 5.1: Diagrama de l'arquitectura de Sauron

El sistema organitza la informació per a la gestió de la següent forma:

- ***servers***: És l'element base de la jerarquia. Permet definir les opcions globals de BIND i DHCP i generar-ne les configuracions.
  - ***zones***: Un server pot contenir un conjunt de zones.
    - ***hosts***: Una zona pot contenir un conjunt de *hosts*.
  - ***nets***: Un server permet definir un mapa de topologia de xarxa nivell 2 (VLANs) i de nivell 3 (subnets).

La figura 5.2 mostra un arbre de la jerarquia d'aquests elements.

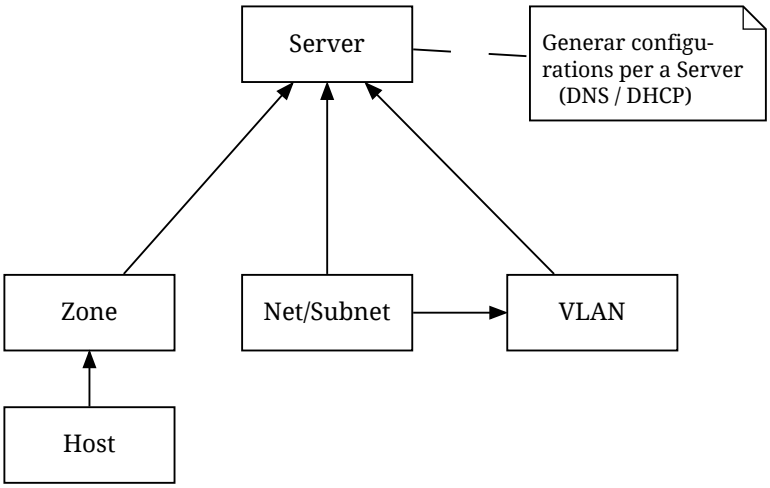

Figura 5.2: Organització dels elements de gestió de Sauron

A partir de les dades de zona, hosts i topologia per a cada server que disposi el sistema, Sauron genera dinàmicament les configuracions per a BIND i DHCP.

5.3 Anàlisi FLOSS del projecte

Sauron és un projecte FLOSS (*Free/Libre Open Source Software*). Per les seves característiques es cataloga com a projecte tipus *Solo* segons la caracterització que es dóna en [Rib09]. Aquest tipus de projectes es caracteritzen per ser projectes reduïts en els que habitualment una sola persona és responsable del projecte sencer.

Per tal que la comunitat d'usuaris pugui comunicar-se tant amb els desenvolupadors com entre la pròpia comunitat per aportar *feedback*, reportar problemes, resoldre incidències, etc., Sauron disposa de diverses llistes de correu públiques:

- sauron-users@lists.jfi.yu — discussió general i suport
- sauron-devel@lists.jfi.yu — desenvolupament i *bug tracking*
- sauron-announce@lists.jfi.yu — anuncis de noves versions

A continuació es descriu de forma formal el projecte Sauron segons els models [BLM08], [RGC03].

5.3.1 Tarjeta d'identitat

La targeta d'identitat conté les dades i característiques bàsiques i essencials del projecte. Aquestes es mostren a continuació.

Nom Projecte	Sauron
Repositori	http://sauron.jyu.fi/pub/sauron/
Pàgina Web	http://sauron.jyu.fi/
Descripció	Proporcionar un sistema escalable de gestió de *hosts* i servidors DNS/DHCP.
Llicència	GNU GPLv2
***Standalone*/Part d'un altre projecte**	Standalone
Tipus de Software	Gestió de servidors
Organització	Universitat de Jyvaskyla
Sistema Operatiu	GNU/Linux
Cost per l'usuari	Gratuït
Equip desenvolupador	Timo Kokkonen i personal del centre de computació de la Universitat de Jyvaskyla
Mida Comunitat	Reduïda (< 51).
Llenguatges	Perl, bash.
Eines de gestió	Control de versions: CVS Bug tracking: No Listes de correu: sauron-users, sauron-devel, sauron-announce
Primer alliberament	Gener de 2003
Filosofia Disseny	Proporcionar un mecanisme basat en permisos d'usuari/grup per a la delegació de la gestió dels servidors DNS/DHCP.
Darrera versió estable	Versió 0.6.2 – 18 de juny de 2003
Darrera versió desenvolupament	Versió 0.7.3 – 18 de febrer de 2010
Lider del Projecte	Timo Kokkonen – Universitat de Jyvaskyla
Finançament	Universitat de Jyvaskyla

5.3.2 Anàlisi del repositori

A través de l'anàlisi del repositori es poden extreure dades de l'evolució del projecte al llarg del temps i realitzar estimacions del cost i esforç de desenvolupament del mateix. El codi del projecte està disponible per a l'accés públic de lectura en un repositori CVS.

Mitjançant les eines Sloccount i CVSAnalY s'han extret les dades presentades a continuació.

Anàlisi sloccount

Sloccount és una eina que realitza un anàlisi de les línies del codi i en mostra alguns resultats, com els llenguatges utilitzats en els diferents mòduls del projecte. Aquesta eina fa ús del model COCOMO[3] per tal de realitzar una estimació del cost, esforç i temporització d'un projecte.

Sauron, per les seves característiques, i segons aquest model, es tracta d'un projecte de tipus *Organic* i s'estableixen els paràmetres factor (f) i exponent (e) per a l'esforç (E) i la temporització (S) respectivament, als valors següents, tal i com indica el model:

- $E_f = 2.4$
- $S_f = 2.5$
- $E_e = 1.05$
- $S_e = 0.38$

Amb aquests paràmetres i amb l'ús de Sloccount es determina l'esforç en Persones-Any necessàries per al desenvolupament del projecte i el temps de desenvolupament. Aquestes dades es mostren en la taula 5.2. Per al calcul econòmic s'han ajustat els valors salarials a $32.000 anuals i un *overhead* del 40%.

Línies totals de codi:	20.678
Esforç de desenvolupament, persones-any (persones-mes):	4.81 (57.74)
Estimació de temps, anys (mesos):	0.97 (11.68)
Nombre de desenvolupadors estimat:	4.95
Cost econòmic total estimat:	61.592 $

Taula 5.2: Resultats del model COCOMO en l'anàlisi del projecte

[3] COCOMO és un model per al càlcul dels costos de desenvolupament d'un projecte a través de les línies de codi.

La taula 5.3 mostra un resum de les linies de codi de tot el projecte.

Perl	18119 línies (87.62%)
Bash	2558 línies (12.37%)

Taula 5.3: Resum de les linies de codi i llenguatges emprats

Tal i com s'observa en la taula 5.2 el temps estimat de desenvolupament s'estima en 0.97 anys a raó de 4.95 desenvolupadors. L'esforç de desenvolupament s'ha estimat en 4.81 PA[4], que equival per tant a 4.81 vegades el treball d'una persona en un any. Tenint en compte doncs, que el projecte ha estat desenvolupat per una sola persona, s'obté,

$$\frac{4,81 \text{ persones} \cdot \text{any}}{1 \text{ persona}} = 4,81 \text{ anys},$$

valor que sembla ajustar-se amb força exactitud al temps de desenvolupament actiu real tal i com s'ha comentat en la descripció general (1999—2004).

5.3.3 Anàlisi CVSAnalY

CVSAnalY és una eina d'anàlisi estadístic de repositoris CSV (també amb suport parcial per a Subversion) desenvolupada en el si d'un projecte de recerca en programari lliure de la Universitat Rey Juan Carlos de Madrid. Aquesta utilitat genera una base de dades local amb la informació extreta del repositori i n'analitza les dades i *logs* del repositori per tal de generar estadístiques i gràfics de l'activitat dels *commiters*.

D'aquesta forma s'ha pogut reportar que l'únic *commiter* del projecte es correspon al seu autor. Per tant ningú altre ha contribuït codi de forma directa al repositori. A continuació es mostren algunes de les dades i dels gràfics que s'han considerat més rellevants.

- Primer commit al repositori: 2000-03-09 16:31:03
- Darrer commit al repositori: 2010-05-01 07:22:32
- Nombre de commits: 1920
- Nombre de commiters: 1

En la figura 5.3 es pot observar com al voltant de la setmana 180 es produeix una reducció en el nombre de lnies de codi, probablement deguda al re-disseny/actualització d'algun mòdul. Aquest fet coincideix amb l'alliberament de la versió 0.7.0 (juliol de 2003).

[4]Persones-Any, no confondre amb persones per any (pers./any)

La figura 5.4 mostra com la tendència del nombre de commits canvia i s'estanca a partir de la setmana 200 (final del 3r any), any en el que l'autor abandona el seu lloc al centre de computació (2004).

La figura 5.5 mostra com el nombre de fitxers passa del voltant de 170, a 200, poc després de la setmana 200, que correspon a principis de 2004; fet que coincideix amb l'alliberament de la versió 0.7.1 (gener de 2004).

En les figures 5.3 i 5.4 s'observa com el nombre linies de codi i de commits torna a créixer al voltant de la setmana 255, entre 3 i 4 mesos abans de l'alliberament de la versió 0.7.2 (abril de 2005).

En la figura 5.3 s'observa com el nombre de línies de codi s'ha mantingut pràcticament estancat des de la setmana 270 (maig de 2005).

En l'histograma de la figura 5.6 s'observa que el major nombre de commits es va produir en diumenge, i que aproximadament un 30% del total de commits es va produir entre dissabte i diumenge; el qual fa pensar que el projecte es desenvolupava també com a projecte personal fora de l'horari laboral al centre de computació.

L'histograma de la figura 5.7 mostra com la major part de commits s'han realitzat entre les 20 i les 21 hores.

5.4 Modificacions realitzades

Algunes característiques de Sauron han estat corregides o millorades en el marc d'aquest treball. En l'apèndix A es descriuen les correccions i millores que s'han realitzat sobre el codi de Sauron. Concretament les secccions A.3.1 i A.3.2 descriuen els *bugs* corregits i les funcionalitats menors corregides/afegides, respectivament. En l'apèndix C es llisten els pegats generats en en base a aquestes modificacions en format GNU diff unificat.

L'objectiu és enviar a la comunitat de Sauron tant els pegats com els apèndix A–D per tal de contribuir-hi tant amb codi com amb documentació donat que les darreres versions presenten problemes amb les noves instal·lacions i que algunes de les funcionalitats del sistema com per exemple la gestió de la topologia de xarxa per tal d'obtenir configuracions de DHCP funcionals, no estan prou documentades; o l'existència d'alguns *bugs* que afecten tan a la importació de dades com al maneig de la interfície.

En aquests apèndix també es descriuen configuracions avançades per al sistema DNS que es poden integrar amb Sauron a través dels *scripts* escrits per al propòsit i que poden ser d'interès per a la comunitat.

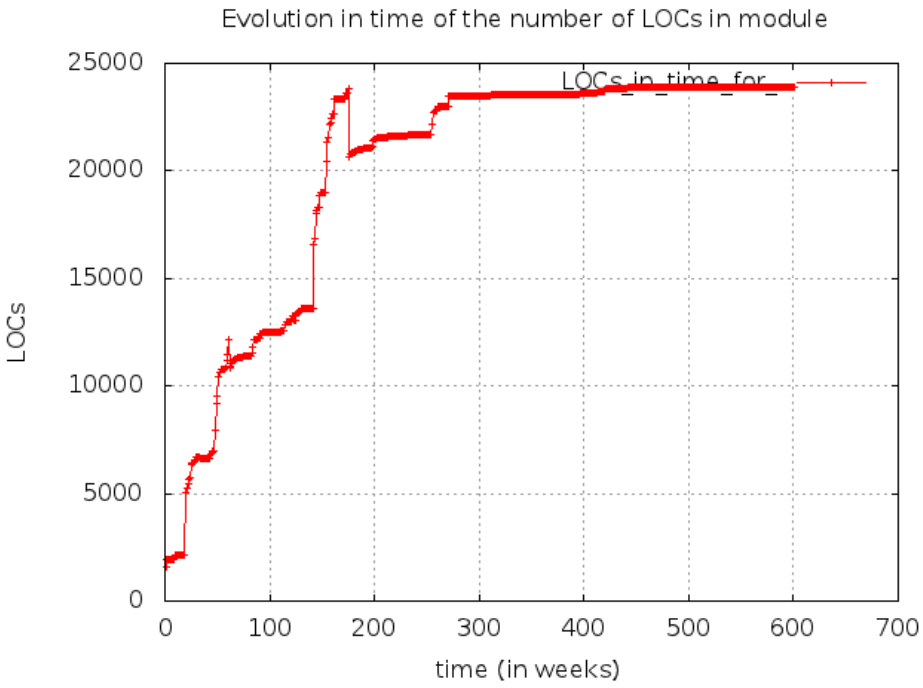

Figura 5.3: Evolució del nombre de ínies de codi (LOCs)

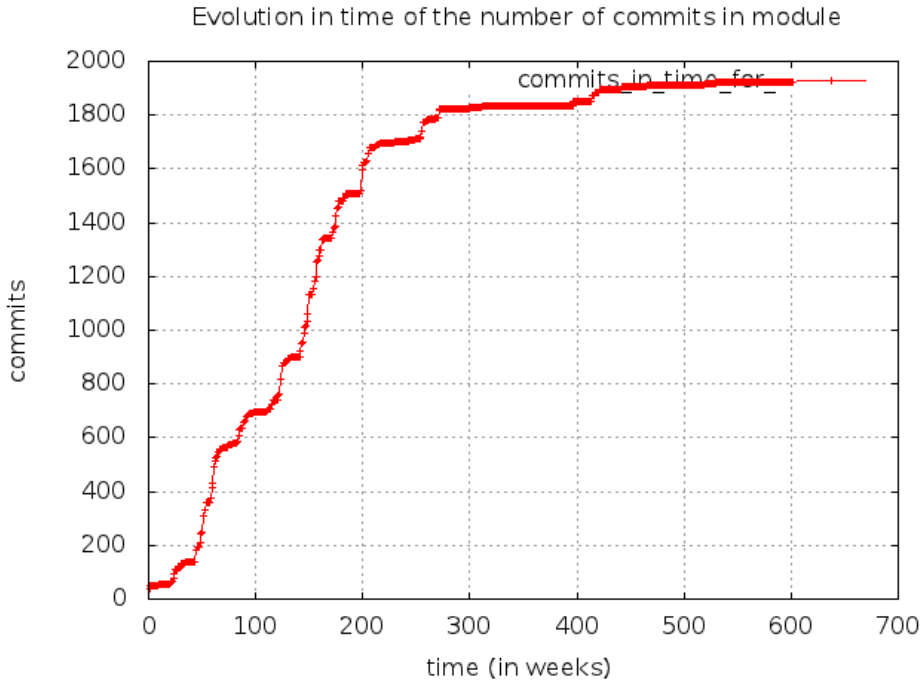

Figura 5.4: Evolució del nombre de commits

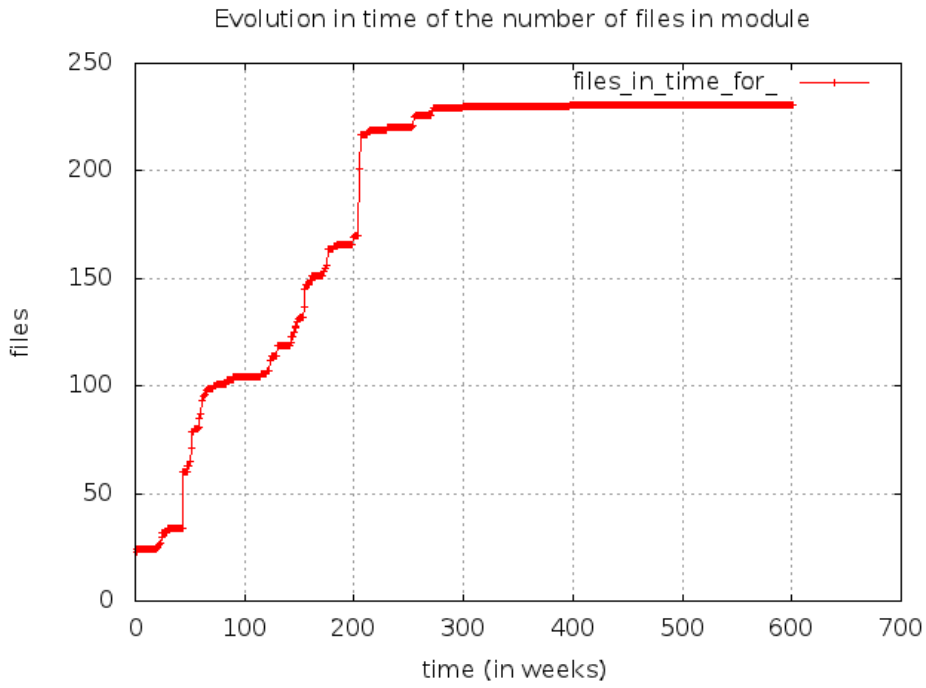

Figura 5.5: Evolució del nombre de fitxers

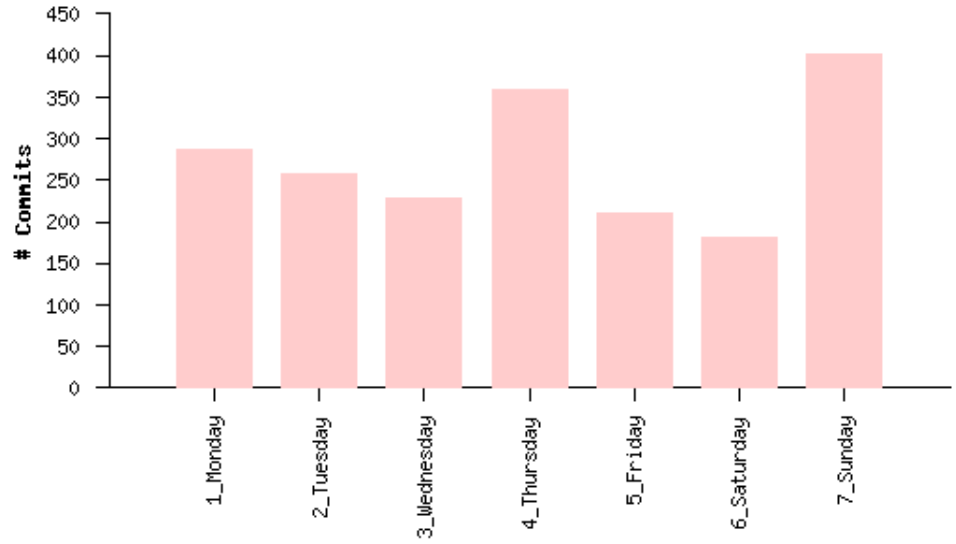

Figura 5.6: Histograma de commits per dia de la setmana

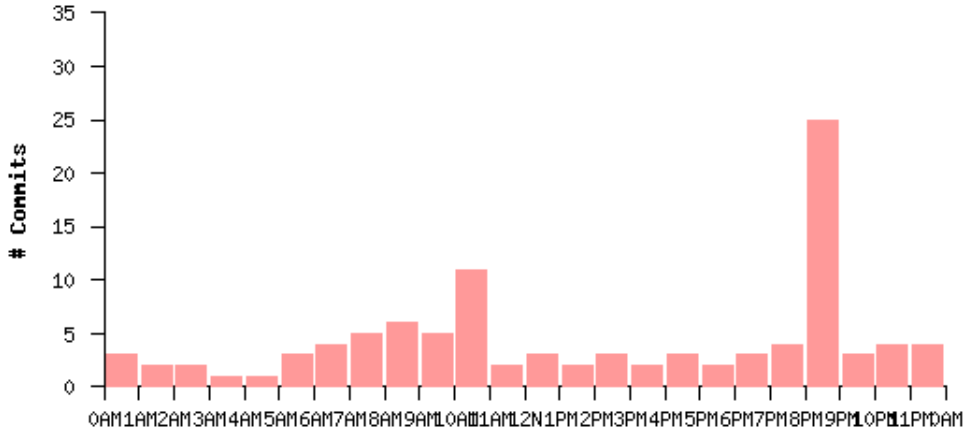

Figura 5.7: Histograma de commits segons franja horària

Capítol 6

BIND *dynamic backend*

En aquest capítol es mostren algunes de les extensions disponibles per a BIND. Com ja s'ha vist en el capítol 2 —on s'ha introduït el servidor de noms BIND—, BIND llegeix les dades de les zones de fitxers de text i les manté en memòria durant l'execució per tal d'atendre les peticions que rep. Poder disposar de les zones del DNS emmagatzemades en una base de dades aporta alguns avantatges, entre altres:

- Càrrega dinàmica de les dades de zona.
- S'elimina la necessitat de recarrega dels serveis quan es produeixen canvis.
- Permet desfer-se els servidors esclaus i la transferència de zones.
- Permet desfer-se dels fitxers de zona que tendeixen a tornar-se ≪enrevessats≫ per a zones grans i/o mantingudes per diversos administradors.
- permet gestionar les dades de zona a través d'aplicacions o *scripts* externs.
- Brinda un mecanisme de control d'accés i modificació de les dades de zona.

6.1 APIs de BIND

El servidor ISC BIND disposa de dues APIs:

- La primera (≪*Advanced*≫ *Database API*) es una API avançada disponible des de la versió 8 de BIND i que permet substituir el *backend* en memòria de BIND a través de rutines escrites per l'usuari. L'ús d'aquesta API però no és una tasca trivial donat que totes les funcionalitats del protocol DNS han de ser suportades per les rutines escrites per l'usuari. Tot i que aquesta API proporciona suport complet a les funcions de BIND, la documentació sobre aquesta és pràcticament inexistent i es redueix als comentaris del fitxer de capçalera de C (`db.h`).

- La segona API (*Simple Database API*) proporciona una interfície per a l'ús de *data sources* alternatius per a l'emmagatzemament de les dades.

Cap d'aquestes APIs permet carregar zones de forma dinàmica i per tant cal recarregar la configuració de BIND (rndc reload) per tal que llegeixi de nou els fitxers de configuració (named.conf) amb la declaració de les zones.

Aquestes APIs estan enllaçades estàticament en el codi de BIND i per tant no es permet la càrrega dinàmica de mòduls. Per tal d'incloure el suport de les llibreries escrites per l'usuari, caldrà recompilar les fonts de BIND incloent-hi les llibreries a carregar.

Una tercera API (*Dinamic Loadable Zones)*[1] es va desenvolupar el 2002 sota els termes de la llicència BSD per la fundació NLnet[2], i distribuïda inicialment com a pegat de BIND. Aquesta API permet tal i com el nom indica la càrrega dinàmica de noves zones sense la necessitat d'aturar el servei. A més incorpora diversos *drivers* per tal d'utilitzar diferents fonts de dades per a l'emmagatzemament, entre els quals, *File system, LDAP, Postgres, MySQL,...*

6.1.1 Simple Database API (sdb)

Aquesta API està disponible des de la versió 9.1 de BIND. Es tracta d'una abstracció consistent en cinc funcions de *callback* i un conjunt de funcions per a manejar els registres de DNS (RR). Aquesta API permet escriure *drivers* per a proporcionar diferents fonts de dades per a l'emmagatzemament de les zones (e.g. una base de dades relacional o un directori LDAP), així com manipular les dades de zona mitjançant algun algorisme definit per l'usuari (e.g. implementar un balanceig de càrrega). A l'igual que l'ADB API, la documentació d'aquesta es redueix als comentaris d'un fitxer de capçalera de C (`sdb.h`) tot i que una descripció global i alguns exemples d'ús es troben disponibles en [Ait05].

El pas de paràmetres a la interfície que proporciona aquesta API es realitza a través del fitxer de configuració de BIND `named.conf` mitjançant la directiva `database`. Aquesta directiva s'utilitza en la declaració de les zones i substitueix la directiva `file`.

Les cinc funcions de *callback* que ofereix aquesta API es llisten a continuació:

- **create()**: Es crida quan una zona és inicialitzada per BIND. Per tant es crida en cada instància del driver declarada al fitxer `named.conf`.

[1] http://bind-dlz.sourceforge.net/license.html

[2] NLnet és una fundació Neerlandesa originada el 1982 per tal de desenvolupar i proveir serveis de xarxa a Europa. http://www.nlnet.nl/

- **destroy()**: Es crida quan una zona es descarrega de BIND (e.g. al tancar una connexió amb la BD).
- **lookup()**: Es crida en la recepció d'una petició (*o query*). El driver proporcionarà el resultat per tal d'incloure'l en la resposta.
- **authority()**: Es crida en cas que la funció lookup() no retorni els registres SOA i NS. Realitza una petició dels registres SOA i NS per tal d'emplenar la secció ≪Authority≫ en la resposta d'una petició DNS.
- **allnodes()**: Es crida en iniciar una transferència de zona (AXFR). Retorna tots els registres de la zona.

6.2 Backend dinàmic per a BIND

Existeix un driver basat en l'API SDB, que permet fer ús d'un directori LDAP com a *backend* per a BIND. Aquest driver fou originalment escrit per Stig Venaas, actualment membre de l'IETF, com un experiment en el seu temps lliure el 2002 mentre treballava a UNINETT[3], per tal de provar la nova interfície de BIND (l'API SDB). Venaas s'adonà que no existia cap esquema per a LDAP per emmagatzemar la informació de les zones de BIND, de manera que va crear l'esquema dNSZone. Aquest esquema es pot consultar a la secció D.7 de l'apèndix D.

L'esquema proporciona un mapeig entre els fitxers de zona de BIND i el directori.

Quan s'utilitza aquest driver, el servidor de noms realitza una consulta al directori per a cada petició que rep i deixa de realitzar *caching*. Una correcta configuració i indexat del servidor OpenLDAP permet fer un ús eficient de la *caché* de la base de dades *Berkeley Database* que OpenLDAP utilitza de *backend*, reduïnt d'aquesta manera les operacions d'E/S a disc i proporcionant un rendiment òptim.

L'esquema de l'arquitectura i el desplegament d'aquesta solució es troben descrits en l'apèndix B. Tal i com es descriu en aquest apèndix, es pot fer ús de la replicació del directori per tal de mantenir cada servidor de noms actualitzat sense necessitat de servidors esclaus ni transferència de zones.

L'ús d'aquest driver permet flexibilitzar la configuració del sistema DNS. Per exemple es pot optar per utilitzar aquest el driver només per a determinades zones o només per algun servidor i seguir utilitzant simultàniament fitxers de zona, servidors esclaus i transferència de zones.

La versió 1.0 presenta alguna restricció en l'accés a LDAP. Concretament només permet la connexió a través de *sockets TCP*. Per tant la comunicació a través de l'ús de

[3]Uninet és una companyia del govern de Noruega responsable del manteniment d'una xarxa de comunicacions nacional dedicada la recerca i connectada a altres xarxes del nord d'Europa formant part d'Internet.

sockets UNIX mitjançant l'esquema «ldapi://» no és possible. L'ús de *sockets* UNIX representa una millora del rendiment donat que estalvia l'*overhead* que implica l'ús del protocol TCP/IP. Tot i això el rendiment proporcionat es considera suficient. De la mateixa forma l'ús d'una connexió sobre SSL/TLS mitjançant «ldaps://» tampoc està suportat. La branca experimental 1.1 del driver afegeix aquestes funcionalitats.

6.3 Paquet bind-sdb

Existeix un paquet disponible als repositoris de CentOS que incorpora suport per al driver de LDAP descrit en la secció 6.2 i per al mòdul DLZ.

Aquest paquet proveeix d'un conjunt de drivers junt a la versió recompilada del servidor i que mitjançant l'API SDB permeten carregar les dades des de diverses fonts de dades, algun d'aquests controladors però (com el controlador Postgres o el SQLite), són experimentals i no se n'aconsella l'ús en entorns de producció. El controlador per a LDAP s'ofereix com a estable i per tant s'ha escollit fer-ne ús com a *backend* d'emmagatzemament.

6.3.1 Utilitats LDAP i syncldapzone

El paquet bind-sdb proveeix també de dues utilitats amb la finalitat de **carregar** les dades dels fitxers de zona de BIND en un directori LDAP:

- la primera, es tracta d'un programa escrit en C que actualitza directament el directori a través de les llibreries de LDAP;
- la segona, un script en Perl que genera un fitxer LDIF[4] amb les dades de la zona. Bàsicament es tracta d'un *dump* de la zona en format apte per a LDAP.

  Aquest script presenta però un problema: llegeix el fitxer de zona d'una passada i escriu l'LDIF a mesura que llegeix. D'aquesta forma, en el cas –que per la forma com Sauron genera els fitxers es dóna– que un host tingui diversos registres (RRs) separats en diferents linies (e.g. dos registres «A» separats), el fitxer l'LDIF generat tindrà diversos «Distinguished Name» repetits.

  Això provocarà que es produeixi un error a l'afegir les dades del fitxer al directori, donat que el DN ha de ser únic.

A més a més, *tots dos programes presenten un clar problema*: només carreguen noves dades al directori i no poden actualitzar entrades ja existents. Per tant, podrien servir per realitzar la càrrega inicial, però no permeten actualitzar el directori un cop

[4] *LDAP Interchange Format*, és el format d'intercanvi de dades de LDAP.

generat. Per tant, cal un mecanisme per tal que els fitxers de zona de Sauron/BIND puguin mantenir actualitzat el directori.

Així doncs s'ha optat per escriure un programa que solucioni el problema de forma efectiva i permeti mantenir el directori actualitzat amb les dades proveïdes per Sauron. Una descripció detallada d'aquest mecanisme es troba en la secció 8.1.1.

Capítol 7

Entorn de simulació i *testeig*

En aquest capítol es descriu l'entorn utilitzat durant el desenvolupament del projecte per tal de configurar un entorn de simulació on poder desplegar i avaluar el funcionament del sistema que es descriu en els apèndix A i B. Aquest entorn es compon de diversos servidors virtualitzats i els corresponents enllaços de xarxa entre aquests.

7.1 Entorn virtual

Per tal de dur a terme aquesta simulació, s'ha configurat un entorn de treball basat en màquines virtuals (*o VMs*), emulant cadascun dels servidors que actualment corren sobre màquines físiques. Aquestes màquines virtuals (*guests*) corren sobre una màquina física amfitrió (*host*).

Mitjançant la virtualització ha estat possible la instal·lació dels servidors de noms, del servidor DHCP i del servidor per a Sauron. Així com la verificació de la correcta configuració i funcionament dels serveis simulant la pròpia estructura dels servidors físics actualment en producció; configurant d'aquesta forma un entorn virtual què respon a la pròpia arquitectura de xarxa de la universitat. Aquest entorn de proves es descriu en més detall al llarg aquest capítol.

A més a més, disposar dels servidors virtualitzats, permetrà moure directament[1] aquestes màquines al nou sistema de virtualització de la universitat i posar els serveis en producció de forma immediata, substituint els actuals servidors físics; col·laborant així en l'actual procés de migració de màquines físiques.

En el capítol 4 s'han descrit els servidors que prenen part en aquest projecte i les modificacions proposades en el disseny actual per tal de simplificar-lo lleugerament:

[1] Exportar les VMs en format *Open Virtualization Format (OVF/OVA)* permet que aquestes siguin portables a altres hipervisors com ESX de VMWare.

- **servidors de noms**:
  - {dns1, dns2, dns3}.udl.net (interns)
  - {gardeny, marraco}.udl.cat (externs)
- **servidor DHCP**:
  - dns3.udl.net

A banda d'aquests servidors ja existents, s'ha instal·lat un servidor addicional encarregat de la gestió dels anteriorment citats. Aquest nou servidor executarà el programari Sauron i gestionarà tant els servidors de noms com el DHCP. En l'apèndix A se'n detalla el procés d'instal·lació i configuració, mentre que la secció B.1.1 de l'apèndix B es dóna una visió general de l'arquitectura global del sistema DNS.

Tots els servidors descrits, tenen instal·lat un sistema operatiu CentOS 6.0.

L'esquema de la figura 7.1 mostra l'estructura d'aquest entorn de proves. Es tracta d'un esquema físic i lògic a nivell de xarxa de l'entorn configurat per tal de dur a terme les proves.

Consideracions sobre l'esquema

- Per tal de simplificar l'esquema, es representen només els servidors Sauron, DNS3 i Gardeny. DNS1 i DNS2 responen a una configuració similar a DNS3 però eliminant-ne la interfície de la VLAN «aules».
- Opcionalment Gardeny pot realitzar la funció de DNS1, atenent també peticions DNS a través de la interfície interna i fent ús de les vistes de BIND per controlar l'accés a les dades. D'aquesta forma en funció de per quina interfície arribi una petició, es respondrà amb una IP interna (privada) o externa (pública).

  Per tal de no modificar l'actual esquema d'adreces dels servidors, es podria afegir una tercera interfície virtual (quarta en l'entorn de proves) associada a la VLAN «intranet» corresponent a DNS1 (10.69.1.1).
- DNS3 està connectat a les dues VLANs ja que funciona com a servidor de noms de la xarxa «intranet» i també de la xarxa «aules». Aprofitant que aquesta màquina te interfícies connectades a totes dues xarxes, executa també el servei de DHCP per a les dues.

Comentaris sobre l'esquema

(i) Les interfícies virtuals estan configurades en mode *bridge* i associades a les interfícies Ethernet físiques de la màquina amfitrió tal i com s'indica en la figura.

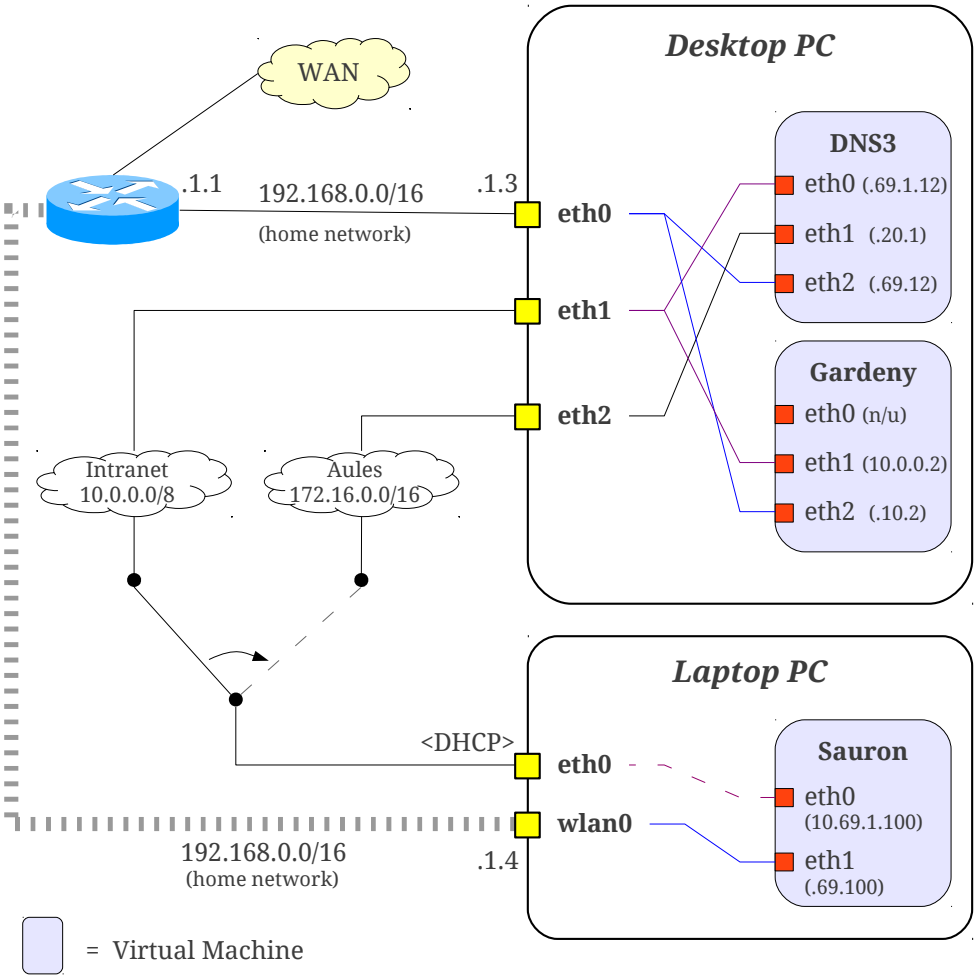

Figura 7.1: Esquema de l'entorn de simulació

Aquest mode permet que la màquina *guest* tingui accés directe a la interfície física i per tant pugui capturar i injectar paquets a xarxa directament.

(ii) Els enllaços de color blau: eth2 (DNS3), eth2 (Gardeny), eth1 (Sauron); i les interfícies associades són exclusivament part de l'entorn de proves. Connecten les màquines *guest* a la xarxa existent (home network) i s'utilitzen per a la gestió de les VMs i per dotar-les de connectivitat a Internet.

Aquestes interfícies es corresponen a la darrera interfície de cada VM, i en el desplegament real seran eliminades, ja que tant la gestió com l'accés a Internet es produeixen a través de la VLAN «intranet».

(iii) Les interfícies eth1 i eth2 de Desktop estan connectades a les VLANs «intranet» i «aules» i en aquesta simulació s'utilitzen per tal de comprovar la cor-

recta configuració i funcionament del servidor DHCP. En aquest cas la màquina Laptop actua com a client i rep les configuracions de xarxa corresponent d'acord a la VLAN que es connecta.

(iv) La interfície eth0 de Laptop s'utilitza tant per comprovar el correcte funcionament del servidor DHCP, utilitzant Laptop com a client; com per connectar Sauron a la xarxa «intranet».

(v) En la simulació, la interfície eth0 de Sauron es pot emprar per a la comunicació amb la resta de servidors quan aquesta està connectada a la xarxa «intranet», però es pot prescindir d'aquest enllaç en les proves donat que es disposa de la xarxa 192.168.0.0 per a la gestió, accessible a través de eth1 de Sauron.

(vi) Les interfícies eth0 (Desktop) i wlan0 (Laptop) són les que utilitzen les màquines *host* en la seva operació habitual, per tant són compartides amb les màquines *guest*.

(vii) La comunicació entre una màquina *host* i els *guests* que corre, es realitza a través de la xarxa física degut a la configuració en *bridge*. Per tant, en aquesta configuració, encara que els paquets tornin per la interfície per on han sortit, és necessari un switch per encaminar-los.

(viii) La interfície eth0 de Gardeny no s'utilitza en les proves. Es correspon a la interfície pública del servidor DNS.

7.1.1 Taules d'encaminament

Tot seguit es mostren les taules d'encaminament per als tres servidors de l'esquema de la figura 7.1.

Destí	**Gateway**	**Màscara**	**Interfície**
192.168.0.0	0.0.0.0	255.255.0.0	eth1
10.0.0.0	0.0.0.0	255.0.0.0	eth0
0.0.0.0	192.168.1.1	0.0.0.0	eth1

Taula 7.1: Taula d'encaminament de Sauron

Destí	**Gateway**	**Màscara**	**Interfície**
172.16.0.0	0.0.0.0	255.255.0.0	eth1
192.168.0.0	0.0.0.0	255.255.0.0	eth2
10.0.0.0	0.0.0.0	255.0.0.0	eth0
0.0.0.0	192.168.1.1	0.0.0.0	eth2

Taula 7.2: Taula d'encaminament de DNS3

Destí	Gateway	Màscara	Interfície
192.168.0.0	0.0.0.0	255.255.0.0	eth2
10.0.0.0	0.0.0.0	255.0.0.0	eth1
0.0.0.0	192.168.1.1	0.0.0.0	eth2

Taula 7.3: Taula d'encaminament de Gardeny

7.1.2 Serveis i ports d'escolta

A continuació s'indiquen els serveis que executa cada màquina i es mostra la sortida de `netstat` per veure'n els ports d'escolta.

DNS3

- *Serveis*: slapd (OpenLDAP), named-sdb (BIND), sshd (OpenSSH), dhcpd (DHCP)

Proto.	Adreça local	Adreça remota	Estat	Servei
tcp	0 0.0.0.0:389	0.0.0.0:*	LISTEN	slapd
tcp	192.168.69.12:53	0.0.0.0:*	LISTEN	named-sdb
tcp	172.16.20.10:53	0.0.0.0:*	LISTEN	named-sdb
tcp	10.69.1.12:53	0.0.0.0:*	LISTEN	named-sdb
tcp	127.0.0.1:53	0.0.0.0:*	LISTEN	named-sdb
tcp	0.0.0.0:22	0.0.0.0:*	LISTEN	sshd
tcp	127.0.0.1:953	0.0.0.0:*	LISTEN	named-sdb
tcp	0.0.0.0:636	0.0.0.0:*	LISTEN	slapd
udp	192.168.69.12:53	0.0.0.0:*		named-sdb
udp	172.16.20.10:53	0.0.0.0:*		named-sdb
udp	10.69.1.12:53	0.0.0.0:*		named-sdb
udp	127.0.0.1:53	0.0.0.0:*		named-sdb
udp	0.0.0.0:67	0.0.0.0:*		dhcpd
raw	0.0.0.0:1	0.0.0.0:*	7	dhcpd

Taula 7.4: Ports d'escolta i serveis associats (DNS3)

Gardeny

- *Serveis*: slapd (OpenLDAP), named-sdb (BIND), sshd (OpenSSH)

Proto.	Adreça local	Adreça remota	Estat	Servei
tcp	0 0.0.0.0:389	0.0.0.0:*	LISTEN	slapd
tcp	192.168.69.1:53	0.0.0.0:*	LISTEN	named-sdb
tcp	10.69.1.1:53	0.0.0.0:*	LISTEN	named-sdb
tcp	10.0.0.2:53	0.0.0.0:*	LISTEN	named-sdb
tcp	127.0.0.1:53	0.0.0.0:*	LISTEN	named-sdb
tcp	0.0.0.0:22	0.0.0.0:*	LISTEN	sshd
tcp	127.0.0.1:953	0.0.0.0:*	LISTEN	named-sdb
tcp	0.0.0.0:636	0.0.0.0:*	LISTEN	slapd
udp	192.168.69.1:53	0.0.0.0:*		named-sdb
udp	10.69.1.1:53	0.0.0.0:*		named-sdb
udp	10.0.0.2:53	0.0.0.0:*		named-sdb
udp	127.0.0.1:53	0.0.0.0:*		named-sdb

Taula 7.5: Ports d'escolta i serveis associats (Gardeny)

Sauron

- *Serveis*: slapd (OpenLDAP), sshd (OpenSSH), postmaster (Postgres), httpd (Apache)

Proto.	Adreça local	Adreça remota	Estat	Servei
tcp	0.0.0.0:389	0.0.0.0:*	LISTEN	slapd
tcp	0.0.0.0:22	0.0.0.0:*	LISTEN	sshd
tcp	127.0.0.1:5432	0.0.0.0:*	LISTEN	postmaster
tcp	0.0.0.0:636	0.0.0.0:*	LISTEN	slapd
tcp	:::80	:::*	LISTEN	httpd

Taula 7.6: Ports d'escolta i serveis associats (Sauron)

7.2 Programari de virtualització

Per tal de dur a terme la simulació s'ha utilitzat el paquet Oracle VirtualBox v4. VirtualBox és un paquet de programari de virtualització multiplataforma disponible per diversos sistemes operatius i arquitectures.

VirtualBox es distribueix sota la llicència lliure GPLv2 tot i que existeix un paquet addicional amb components privatius llicenciats sota una llicència PUEL (*Personal*

Use and Evaluation License) que en permet l'ús personal, acadèmic o d'avaluació. Aquest paquet addicional no és necessari per al funcionament del paquet base i inclou algunes característiques com són: suport per a USB 2.0, VirtualBox RDP o suport per al protocol PXE en targes de xarxa Intel.

El format d'emmagatzemament per als discs de les màquines virtuals de VirtualBox és VDI (*Virtual Desktop Image*) però també és compatible amb el format de disc de VMWare (VMDK), permetent així l'intercanvi discs virtuals entre les dues plataformes. VirtualBox també suporta l'estàndard OVF (*Open Virtualization Format*) permetent per tant la portabilitat de màquines entre diferents sistemes de virtualització.

Guest additions

Les *guest additions* proporcionen un conjunt d'utilitats i *drivers* que es poden instal·lar de forma opcional al sistema *guest* i que tenen la finalitat de millorar la integració entre la màquina *host* i la màquina *guest*. Les funcionalitats que ofereixen es centren en un maneig més interactiu de les màquines virtuals i una major integració amb la màquina *host*. Algunes de les característiques que ofereixen aquestes extensions són:

- integració del ratolí,
- ús de carpetes compartides entre *host* i *guest*,
- sincronització de data/hora,
- millora del support gràfic,
- ...

Les guest additions venen incloses a VirtualBox i per instal·lar-les a les màquines *guest* cal muntar una imatge de CD que conté les extensions i executar el binari d'instal·lació. En màquines virtuals Linux, caldrà instal·lar abans però els paquets de desenvolupament del kernel i el compilador de C gcc.

Mitjançant aquestes extensions és pot muntar un directori compartit entre amfitrió i hoste.

Capítol 8

Desplegament del servei

Per tal de fer efectiu el desplegament i l'arrencada del servei cal integrar-ne els diferents components que en formen part, tot adaptant-los a la infraestructura tecnològica existent en universitat. A continuació es descriu en detall com s'ha realitzat la integració i desplegament per dur a terme l'arrencada del servei.

8.1 Desplegament

El sistema a implantar està configurat, de forma global, pels següents components:

(i) diversos servidors de noms (DNS),

(ii) *backend* dinàmic per als servidors de noms (LDAP),

(iii) servidor DHCP,

(iv) servidor Sauron.

En els apèndix A i B, es descriu en detall el procés per al desplegament individual dels components (ii) i (iv). Però per tal de dur a terme el desplegament final i posada en producció del sistema *com un tot*, cal que tots els components quedin integrats i que les modificacions introduïdes a través de la interfície d'usuari de Sauron, es reflecteixin d'una forma efectiva sobre els serveis que gestiona (DNS i DHCP).

Com ja s'ha vist en el capítol 5, Sauron és capaç d'auto-generar els fitxers de zona i configuracions per a BIND i DHCP. Així doncs, a partir de l'escenari descrit en el capítol 7, caldrà algun mecanisme per tal d'executar de forma programada la generació de les configuracions actualitzades i la recàrrega d'aquestes en els serveis corresponents. No només això, sinó que caldrà que aquest mecanisme permeti poblar i mantenir en sincronia el directori LDAP que BIND utilitza com a *backend*, amb

les dades de zona. Per tant, cal que aquest mecanisme gestioni tot el procés de forma automatitzada.

A continuació es descriu la solució desenvolupada per aquest propòsit. Aquesta solució es divideix, tal i com es comenta, en dues tasques: la generació de les configuracions actualitzades i la sincronització d'aquestes amb el directori; a més d'arxivar els resultats i *logs* de tot el procés per a eventuals tasques de *debug* i, col·lateralment, per mantenir còpia de seguretat de les dades tant del DNS com del DHCP.

Aquestes tasques de manteniment corren sobre la màquina Sauron.

8.1.1 Sincronització del directori (syncldapzone.pl)

Descripció

Davant la possibilitat d'utilitzar un directori LDAP com a sistema de *storage* per a les zones del DNS, i la inexistència d'utilitats que atenguessin al propòsit següent de forma efectiva, s'ha escrit un programa en Perl que llegeix i parseja els fitxers de zona de BIND generats per Sauron i en sincronitza les dades al directori, reflectint-hi les modificacions locals de cada fitxer de zona. D'aquesta forma es realitza un *mirroring* dels fitxers contra el directori.

En cas que una zona hagi estat modificada, únicament les entrades corresponents als *hosts* modificats s'actualitzen al directori, reduint d'aquesta forma el trànsit generat, respecte al que suposa la transferència de la zona sencera, davant petits canvis que poden arribar a afectar un únic registre.

Quan una nova zona és creada, aquesta també es carrega al directori, de manera que es pot utilitzar tant per generar el DIT[1] inicial al directori, així com per mantenir-lo en sincronia.

El programa està disponible per a la descàrrega sota la llicència GPLv3, des del següent enllaç:

* http://tfm-mepl.googlecode.com/files/syncldapzone.pl (8.1)

Requisits

Aquest programa fa ús del mòdul de Perl Net::LDAP, per tant cal instal·lar-lo en la màquina que l'executi. Aquest mòdul està disponible a CPAN[2] o a través del gestor de paquets del sistema operatiu, e.g. yum install perl-LDAP, en el cas de CentOS.

[1] S'anomena DIT (*Directory Information Tree*) al conjunt d'objectes jeràrquics del directori LDAP.

[2] CPAN (*Comprehensive Perl Archive Network*), és un repositori de mòduls i documentació per a Perl. http://www.cpan.org

Funcionament

En termes generals, l'*script* realitza les següents operacions:

(i) lectura i parseig dels fitxers de zona de BIND/Sauron;

(ii) construcció d'una col·lecció d'objectes *LDAP::Entry* corresponents a les dades del fitxer llegit, atenent a l'esquema de LDAP *dNSZone*;

(iii) sincronització del directori amb el conjunt d'entrades locals.

Aquest procediment es veu en més detall als diagrames de flux de les figures 8.1, 8.2 i 8.3. Els dos darrers es corresponen a subrutines. Les operacions s'han numerat per tal de poder-ne seguir el fil d'execució.

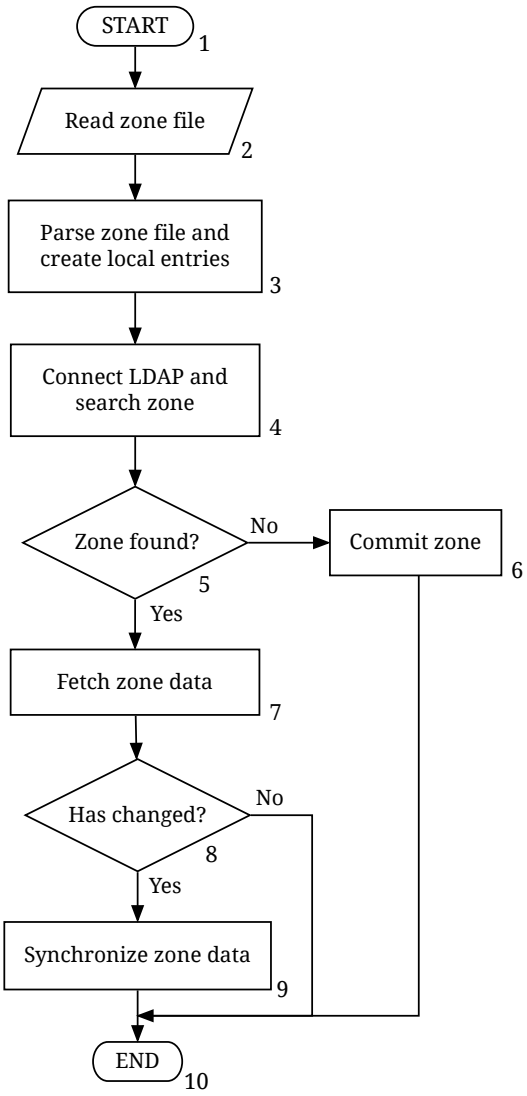

Figura 8.1: Diagrama de flux de `syncldapzone.pl`

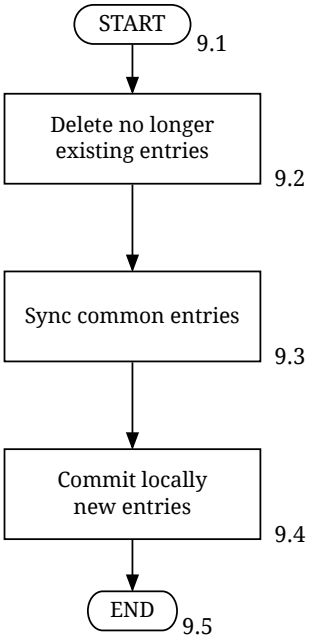

Figura 8.2: Diagrama de flux de l' operació de sincronització de zones

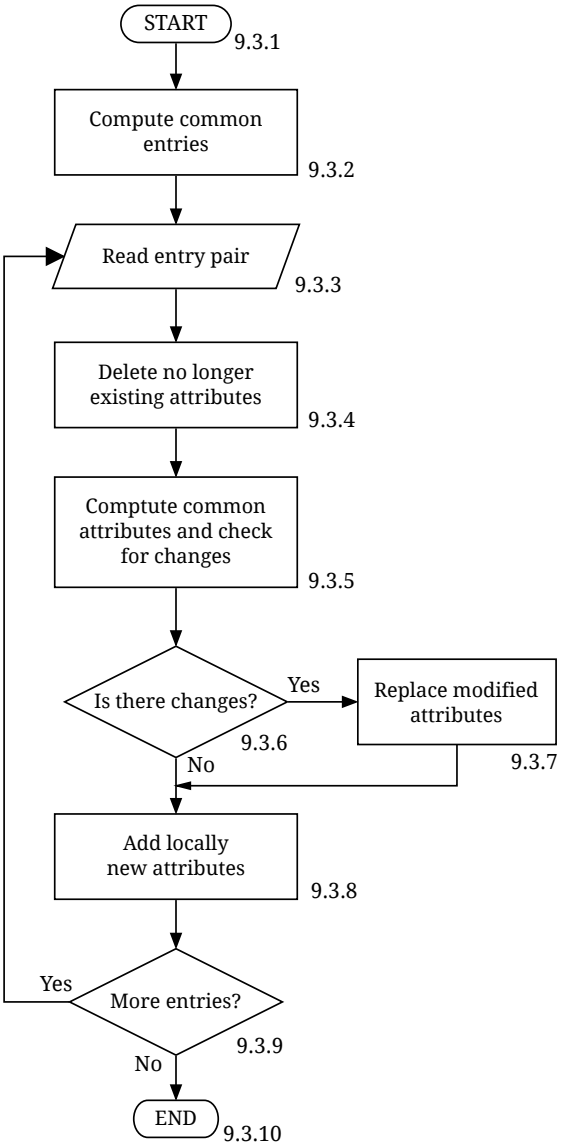

Figura 8.3: Diagrama de flux de l'operació de sincronització d'entrades

8.1.2 Llençador del procés (sauron-launcher.sh)

Descripió

Per tal d'executar tot el procés, s'ha escrit un *script* en Bash que que s'encarrega d'executar Sauron per tal de generar els fitxers de zona i DHCP actualitzats —en cas que hi hagi hagut modificacions. Aquest script realitza les tasques següents:

– execució de Sauron per tal de generar i verificar els fitxers de zona i DHCP actualitzats;
– comprovació del procés de generació;
– copia de les configuracions actualitzades als corresponents *targets* remots;
– generació d'un log de tot el procés;
– arxivament i rotació de les configuracions generades junt amb els logs en un *tarball*.

Algunes variables (*settings*) cal que siguin modificades per l'usuari al seu criteri. La variable `MAXLOGS` determina la rotació de les configuracions antigues.

Aquest script realitza una crida a l'script `syncldapone.pl` descrit anteriorment, per tal de sincronitzar les dades de zona dels fitxers generats amb el directori LDAP.

L'script es pot descarregar des del següent enllaç:

* `http://tfm-mepl.googlecode.com/files/sauron-launcher2.sh` (8.2)

Requisits

(a) Disposar i configurar en l'script, la ruta de l'script `syncldapzone.pl`.

(b) Per tal que l'script pugui connectar-se de forma desatesa als *targets* remots per copiar-hi i recarregar els fitxers de configuració actualitzats, cal configurar autenticació mitjançant claus públiques. En aquest cas s'utilitza SSH i claus públiques RSA.

Funcionament

El funcionament d'aquest script es veu en detall en el diagrama de flux de la figura 8.4.

8.1.3 Programació del procés (cron)

Per tal de propagar les actualitzacions introduïdes a través de la interfície de Sauron, cal l'execució de l'script descrit anteriorment, el qual llença tot el procés. Per tal d'automatitzar aquesta tasca es pot programar el dimoni *crond* per tal que executi periòdicament el procediment.

Programant un temps curt (e.g. 3 – 5 minuts) es garanteix que les modificacions introduïdes a la interfície d'usuari de Sauron es vegin reflectides ràpidament al DNS.

El listing 8.1 mostra l'script de cron que inicia tot el procés.

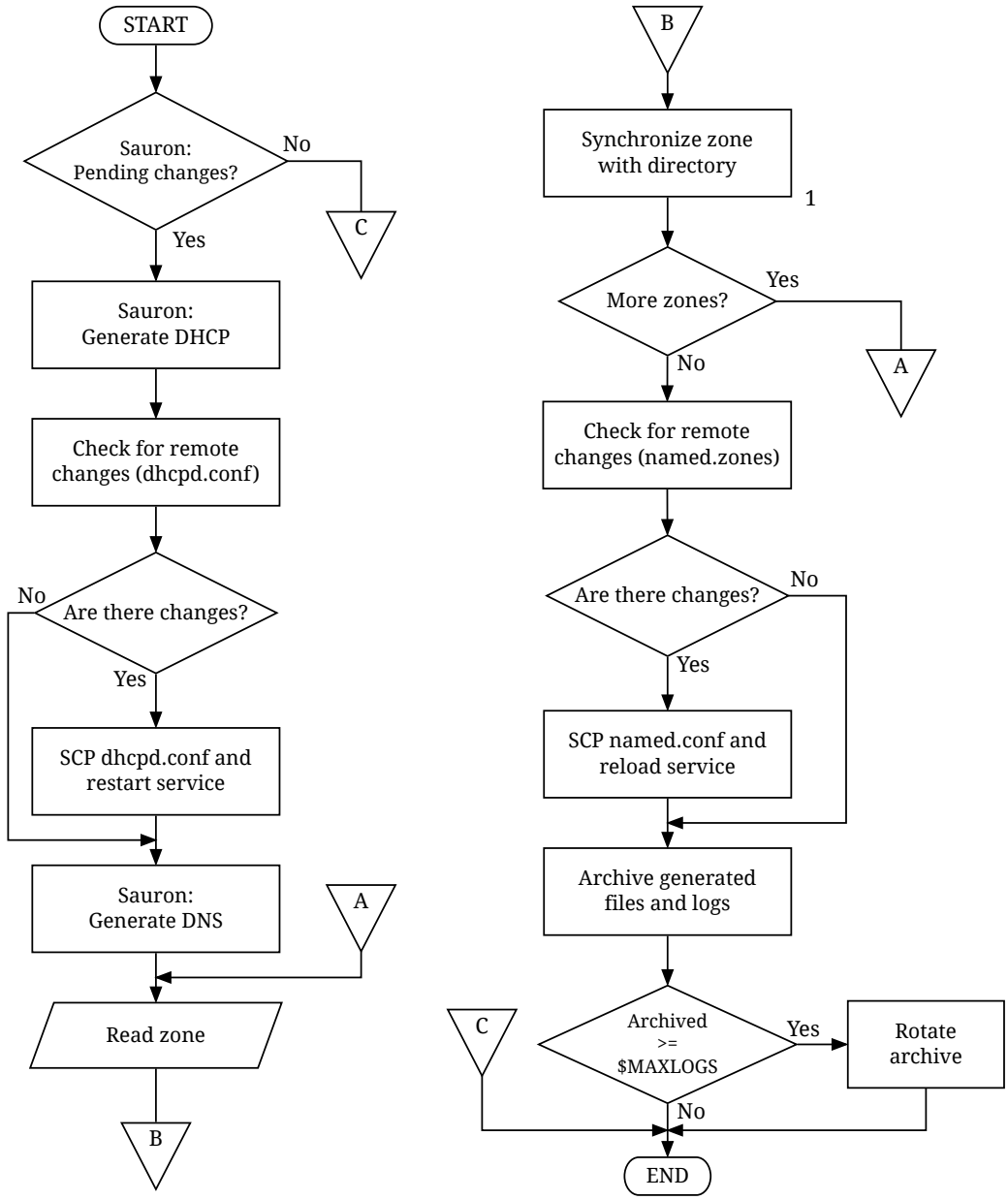

Figura 8.4: Diagrama de flux de `sauron-launcher.sh`

Listing 8.1: /etc/cron.d/sauron

```
# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6)
# |  |  |  |  |
# *  *  *  *  *  command to be executed
sauron_dns1="/root/scripts/sauron-launcher.sh -s dns1 \
                -D cn=Manager,dc=udl,dc=cat -w SECRET \
                -b ou=internal,ou=dns,dc=udl,dc=cat -h 192.168.69.12 \
                192.168.69.1 192.168.69.12"
sauron_gardeny="/root/scripts/sauron-launcher.sh -s gardeny \
                   -D cn=Manager,dc=udl,dc=cat -w SECRET \
                   -b ou=external,ou=dns,dc=udl,dc=cat \
                   193.144.12.130 193.144.8.14"

*/3 * * * * root $sauron_dns1; $sauron_gardeny
```

8.2 Cas UdL. Zones DNS

Sauron ofereix la funcionalitat d'importació de servidors i zones ja existents. Durant la fase inicial de les proves amb Sauron es van advertir alguns problemes en la importació de les zones de la UdL. Aquests problemes es van traduir finalment en *bugs* de Sauron (veieu A.3 per a una descripció detallada).

El primer problema reportat està relacionat amb la interfície de la base de dades i l'ús de *PreparedStatement*, i afecta a la importació de dades. Una descripció detallada es troba en A.3.1.1.

Un altre problema que es va reportar durant la fase de configuració i proves del sistema té relació amb el format dels fitxers de zona que utilitza la UdL, que utilitza noms de host en majúscules per algunes zones que han estat creades automàticament a través d'un script —i que daltra forma hauria passat inadvertit.

La descripció detallada d'aquest *bug* es troba en la secció A.3.1.3 de l'apèndix A.

Aquest bug es va localitzar en el codi posteriorment, i tot i que s'ha corregit, en primera instància i per comprovar-ne el comportament, es va optar per escriure dos senzills scripts (per a les zones internes i les externes) que utilitzant l'editor de *streams* `sed` convertissin tots els noms de host a minúscules; però aprofitant alhora per realitzar

algunes modificacions globals sobre el DNS i corregir alguns problemes en els fitxers atenent a algunes de les recomanacions d'estil per mantenir certa coherència.

Aquestes modificacions es llisten tot seguit.

Zones DNS intern

- conversió dels noms de host a minúscules;
- renombrar el DNS intern (ara dns1): canviar {marraco, marraco.udl.es} per «dns1.udl.net» en els registres SOA i NS de totes les zones;
- modificar la resolució inversa per a que respongui «*.udl.cat» enlloc de «*.udl.es»;
- canviar el camp mail dels registres SOA per «hostmaster.udl.cat» (opcional)

Zones DNS extern

- reemplaçar {gardeny, gardeny.udl.es} per «gardeny.udl.cat» en els registres SOA i NS;
- reemplaçar {marraco, marraco.udl.es} per «marraco.udl.cat» en els registres SOA i NS;
- modificar la resolució inversa: «*.udl.cat» enlloc de «*.udl.es»;
- afegir un registre «NS marraco.udl.cat» en les zones que hi falta;
- canviar el camp mail dels registres SOA per «hostmaster.udl.cat» (opcional)

Aquests scripts es poden descarregar des dels següents enllaços:

* http://tfm-mepl.googlecode.com/files/prepare_files.sh (8.3)
* http://tfm-mepl.googlecode.com/files/prepare_files_gardeny.sh (8.4)

Capítol 9

Conclusions i treball futur

Abans de realitzar una valoració i conclusio d'aquest Treball de Final de Màster, voldria assenyalar que tot i que el sistema presentat no s'ha arribat a posar en producció per causes pressupostàries, el desplegament documentat mostra com el sistema està llest per ser implantat i arrencat, contemplant el context de la universitat i punts dèbils que mereixen especial atenció per tal de realitzar una migració exitosa.

En aquestes pàgines s'ha mostrat la importància del servei DNS i com se'n pot agilitzar la gestió tot millorant-ne i garantint la coherència i correctesa de les dades del mateix; fet d'especial importància donat que repercuteix directament sobre la infraestructura d'Internet per una banda i sobre la infraestructura interna de la universitat per altra banda. Diverses solucions de programari lliure s'han integrat per dur a terme aquest objectiu i s'han realitzat modificacions i millores sobre el codi del sistema Sauron.

L'objectiu dels apèndix A, B, C i D, és no només servir de documentació tècnica d'aquest projecte sinó fer-los públics a la comunitat d'usuaris i desenvolupadors de Sauron, contribuïnt d'aquesta manera al projecte amb documentació i pegats de codi; així com a qualsevol administrador de sistemes que desitgi migrar la gestió manual del seus servidors de noms o fer ús del *backend* dinàmic que LDAP ofereix per al servidor de noms BIND. Fets pels quals aquests apèndix han estat redactats en la llengua anglesa.

Es pot dir que el desenvolupament del treball ha arribat al seu fi amb èxit i que durant el aquest procés s'hi han inclòs i integrat funcionalitats no considerades inicialment. L'ús d'un *backend* dinàmic en el servidor de noms per tal de desfer-se dels fitxers de zona i alhora proporcionar una millor resposta als canvis es va afegir amb posterioritat als objectius d'aquest treball.

Cal esmentar que tot i ser BIND el servidor més desplegat per a la resolució de noms en la infraestructura global d'Internet, la documentació sobre l'ús tant de les APIs que disposa com dels *drivers* disponibles per a fer ús de bases de dades, és escasa i poc mantinguda. D'aquesta manera l'apèndix B aporta documentació actualitzada i

la solució desenvolupada per a la sincronització de dades entre els fitxers de zona i el directori LDAP.

En quant al treball futur una de les vies clarament oberta és el suport d'IPv6 per a Sauron. Algunes discussions entorn al mateix estan obertes i tot sembla indicar que es pot implementar aquest suport de forma «relativament senzilla» sense que això impliqui grans reestructuracions en l'arquitectura. Tant el mòdul de Perl que maneja les adreces com el tipus de dades de la base de dades que Sauron utilitza suporten adreces en format IPv6.

Els resultats obtinguts en la simulació han estat satisfactoris, però caldrà realitzar un *testeig* en un entorn real de producció per avaluar exhaustivament la resposta del servei sota aquesta configuració.

Finalment, en quant a l'aprofitament personal que la realització d'aquest treball m'ha aportat, voldria destacar-ne alguns aspectes com,

- l'estudi del funcionament del sistema de noms de domini i dels servidors de noms;
- l'estudi del funcionament del protocol DHCP;
- l'estudi del funcionament del protocol i directori LDAP;
- l'escarvació de codi per tal de traçar i corregir *bugs*;
- la integració de sistemes, implementació de *scripts*, ús de llenguatges interpretats;
- l'ús metodologies d'anàlisi de projectes FLOSS;
- l'exploració de bibliografia tècnica i especificació d'estàndards;
- la configuració d'entorns de prova virtuals;
- la instal·lació, configuració, arrencada i gestió de servidors virtuals.

Appendix A

Sauron installation and deployment Guide

The following pages show how to deploy Sauron (http://sauron.jyu.fi) a DNS and DHCP management system over a GNU/Linux server.

This guide is based on a GNU/Linux CentOS 6 —but should be valid for any Linux distribution— and on the current Sauron version (0.7.3). Note that the server that host Sauron, does not require graphical environment so all steps from server side of this guide are in command line text–mode (and assuming you are logged as root).

Sauron user interface is web based, so administrators can use it easily and remotely with a simple web browser in a concurrent way. On the other hand some high privilege administration tasks must be made trough command–line interface. A complete list is available at *Sauron User Guide* (*Ch 6. Command Reference*).

Sauron is intended to be a management system for DNS and DHCP servers in a large corporative environments and it can deal with the configurations of the popular BIND and ISC DHCP servers. Required versions:

- **BIND** v8 or later (9.2.x or newer recommended),
- **ISC dhcpd** v2 or later (3.x or newer recommended).

A.1 System pre–requisites

We start at this point assuming a minimal fresh installation of CentOS running in text–mode. Before we can proceed with Sauron, we need to satisfy some system requirements. Table A.1 shows the additional components required by Sauron. This packages can be installed with the distribution's package manager **yum**.

Package	Version	Comments
gcc[2]	-	Required to compile some Perl modules.
perl	≥ 5.6	It is usually installed by default on most Linux systems (already installed).
postgresql-server	$\geq 7.2.3$	Postgres SQL database server.
postgresql-devel	-	Headers and libraries to compile C/C++ code to interact with Postgres DBMS.
httpd	-	Apache web server.

Table A.1: Previous requirements

In order to run, Sauron needs some additional Perl modules. This modules can be installed in several ways, but the Perl CPAN[1] module can be used as shown below. Table A.2 list required modules.

Perl package	Comments
DBI	Database independent interface for Perl. Database access module for the Perl programming language.
Time::HiRes	High resolution time and timers (alarm, sleep, gettimeofday, interval).
DBD::Pg	PostgreSQL database driver for the DBI module.
Net::IP	Perl extension for manipulating IPv4/IPv6 addresses.
Net::DNS	Perl interface to the DNS resolver.
Net::Netmask	Parse, manipulate and lookup IP network blocks.
Digest::MD5	Perl interface to the MD5 Algorithm.
Digest::HMAC	Keyed-Hashing for Message Authentication.
Digest::SHA1	Perl interface to the SHA-1 algorithm.
MIME::Base64	Encoding and decoding of base64 strings.
Crypt::RC5	Perl implementation of the RC5 encryption algorithm.
Net::IDN::Encode	Internationalizing Domain Names in Applications (RFC 3490).
Net::IDN::Punycode	
CGI	Handle Common Gateway Interface requests and responses.

Table A.2: Required Perl packages

Before to proceed with installation, there is another important system-check. You must ensure that your system time is correctly set. The best way to do this is to

[1] CPAN is the Comprehensive Perl Archive Network, a large collection of Perl software and documentation. http://www.cpan.org

[2] **Important:** Remember to remove this package when you finish the installation. A compiler it is not safe for a server!

set up the `ntpd` service to automatically synchronize time from Internet (or your intranet) NTP servers:

```
# yum install ntp
# ntpdate pool.ntp.org
# chkconfig ntpd on
# service ntpd start
```

Note: Be sure your Internet connection is up.

Perl modules installation

Execute the following command to update locate database[2]:

```
# updatedb
```

Now we can proceed to install Perl modules reported in table A.2. To install this modules in an easy way, we proceed with Perl shell provided by CPAN module[3] (see the following example). If installation fails on any module, try to retry (sometimes the first try ends unsuccessfuly):

```
# perl -MCPAN -e shell
cpan[1]> install Net::IP
```

Note: Be sure your Internet connection is up. Replace the example module by each module from table A.2. Press Ctrol+D or type exit once finnished to exit shell.

Answer **no** to manually configuration to autoconfigure modules. If you are asked to install or follow dependencies you must answer **yes**.

PostgreSQL configuration

Finally, the last step is to configure database access. To do this you need to edit `/var/lib/pgsql/data/pg_hba.conf`[4]. PostgreSQL has a client authentication mechanism based on the rules of this file. Each record specifies a connection type, a client IP address range (if relevant for the connection type), a database name, a user name, and the authentication method to be used for connections matching these parameters.

Add the following lines to `pg_hba.conf` in order to allow local access to sauron database from sauron user with md5 ciphered password. (Note that both database

[2] You may need to install `mlocate` package.

[3] You may need to install `perl-CPAN` package using your distribution's package manager.

[4] `pg_hba.conf` is created the first time you start postgresql service, so be sure to start it by simply executing «`service postgresql start`» if the file does not exist yet.

and user names are not created yet; you can choose different names but we are going to use this for convenience):

```
#       DATABASE    USER        CIDR-ADDRESS          METHOD

# "local" is for Unix domain socket connections only
local   sauron      sauron                            md5
local   all         all                               ident sameuser
# IPv4 local connections:
host    all         all         127.0.0.1/32          ident sameuser
# IPv6 local connections:
host    all         all         ::1/128               ident sameuser
```

file: /var/lib/pgsql/data/pg_hba.conf

Note that the position of the new register is important since the first record that match a connection is used to perform authentication: if authentication fails, subsequent records are not considered.

A.2 Sauron installation

Once you have satisfied all system requirements you can proceed with Sauron. As we mentioned in the introduction, at the time to write this guide, the current Sauron release is **0.7.3**, so installation and patching steps covered here are not guaranteed for future releases. Even though this is a beta branch release, users have reported that it is more stable than the stable branch (0.6.x).

We are going to install Sauron from source tarball, so the first step is to download[5] and to uncompress it (you can do that in a temporary directory). Then cd to sauron dir and proceed as shown below:

```
# ./configure
# make
# make docs
# make install
```

*Sauron **has been installed** on `/usr/local/sauron` and `/usr/local/etc/sauron`.*

Now you need to create a database for Sauron. First you must ensure that postgres is running, you must set it up at the start-up scripts to auto-start it each time you boot your system, so you can execute the following:

```
# chkconfig postgresql on
# service postgresql start
```

[5] You can get it from official Sauron project website http://sauron.jyu.fi/download.shtml

And proceed as follows for user and database creation (you must provide a password for that user):

```
# sudo -u postgres createuser -S -D -R -P sauron
# sudo -u postgres createdb -O sauron sauron
```

Sauron made use of Object Identifier (OID) row in database tables, but this is a deprecated feature since PostgreSQL 8.1. So, to enable compatibility with Sauron scripts that populate database tables, to enable OIDs for Sauron database is necessary:

```
# psql -U sauron
sauron=> ALTER DATABASE sauron SET default_with_oids=on;
sauron=> \q
```

config file

The main configuration file is `/usr/local/etc/sauron/config`

You need to set some variables in this file to allow database connection. Edit the following lines with the values of database name, user and password as shown in the example:

```
$DB_DSN = "dbi:Pg:dbname=sauron";
# database user
$DB_USER = "sauron";
# database (user) password
$DB_PASSWORD = "<your user password>";
```

file: /usr/local/etc/sauron/config

Sauron can run in a different machine where your BIND or DHCP servers are hosted, but in order to check BIND and ISC DHCP Sauron–generated configuration files, this packages must be installed (what does not mean that daemons must be running). To enable checking, uncomment the following lines at Sauron config file:

```
# set to enable dhcpd.conf validation from Sauron
$SAURON_DHCP_CHK_PROG='/usr/sbin/dhcpd'
$SAURON_DHCP_CHK_ARGS='-q -t -cf'
# set to enable named.conf validation from Sauron
$SAURON_NAMED_CHK_PROG='/usr/sbin/named-checkconf'
$SAURON_NAMED_CHK_ARGS=''
```

file: /usr/local/etc/sauron/config

Database population

If you have properly set up Sauron to connect database, you can proceed now with database initialization and population. Sauron comes with a set of scripts to do that. First of all you need to create database tables. Then you probably need to update database to the last version[6]. Run `status` script to see if database is at correct version and execute corresponding sql script to update it (if required):

```
# cd /usr/local/sauron
# ./createtables
# ./status
# psql -U sauron -f sql/dbconvert1.3to1.4
```

Now you can follow with global tables (*ethernet manufacturers info* and *root servers*). You can get an updated version of `named.root` from IEEE's website instead of use `contrib/named.root` provided:

```
# ./import-ethers contrib/Ethernet.txt
# ./import-ethers --force contrib/additional-ether-codes.txt
# curl ftp://ftp.rs.internic.net/domain/named.root > contrib/named.root
# ./import-roots default contrib/named.root
```

Before begin using Sauron, you also need to create an administrator account (give the default value `none` for expiration and provide a password for this user). Look at the following output:

```
# ./adduser --superuser
Enter username: admin
Enter group name (empty for none):
Enter user description (full name): Administrator
Enter user email address: admin@fake.ml
Enter optional user info:
Enter account expiration date (dd-mm-yyyy, +<n>d, +<n>y) [none]:
Enter password [JD!JqRGg]: your_password
Username: admin
Group: <none>
Longname: Administrator
email: admin@fake.ml
comment:
expiration: <none>
superuser: true
Add this user [y/n]?y
User admin added successfully.
```

[6] Version 0.7.3 of Sauron requires database at version 1.4.

It is a good idea to make a database dump or backup now before to start importing DNS zones, so we can restore the original one if something goes wrong:

```
# pg_dump -U sauron > ~/sauron-fresh-install.dump
```

At this point Sauron should be ready to operate, but I say should because to get it properly working, ***applying some patches is necessary***. The following section briefly describes some problems that must be corrected in source code and shows how to apply the required patches.

A.3 Patching sources

In order to fix some bugs and add support to some features, additional patches must be applied to some Sauron modules. In this section we will show how to manage to fix it. Described problems has been reported and discussed to `sauron-users` mailing list which is available at the following link: http://lists.jyu.fi/pipermail/sauron-users/

The code of the following patches is listed in appendix C in unified *GNU diff* format and is also available on-line at the following URLs. You have to download and apply it as explained below. You can make use of `curl` or `wget` to get the files or copy them by SSH to your server.

A.3.1 Bug fixing

A.3.1.1 Database interface

Branch 0.7 is making use of Perl DBI module as default instead of former DBD::Pg module. Implementation of Sauron DBI module interface (`DB-DBI.pm`) make use of prepared statements, whose only allows to hold a single SQL statement. But when inserting multiple entries in a table, this module is grouping statements in groups of 25 before to execute it. As a result this will cause problems when importing existing DNS data (see section A.5) because the insertions to DB will fail due tu prepared statement cannot execute a group of statements.

- **Affected file**: `Sauron/DB-DBI.pm`
- **Patch link**: * http://tfm-mepl.googlecode.com/files/DB-DBI.pm.patch (A.1)

A simple patch listed in listing C.1 must be applied to fix this bug. Patch is also available to download from the above link. To apply it proceed as follows:

```
# cd /usr/local/sauron/
# curl PATCH_URL > /tmp/DB-DBI.pm.patch
# patch -p1 -i /tmp/DB-DBI.pm.patch
```

Note: You can simulate patching first by using `--dry-run` option. Remember to back up your files before to proceed.

A.3.1.2 Sauron frontend

The concrete problem is with the menu in web interface to add new hosts (Hosts → Add host). The form to add hosts is experimenting an extrange behaviour and is not shown completely, making impossible to add new hosts, neither for superuser nor any other user.

This bug has been reported as a problem with undefining variables in Perl and the garbage collector. Some discussion is available at the `sauron-users` mailing list[7] 2010. When a variable is undefined, the Perl garbage collector will schedule the cleaning up but this fact can end up in a race-condition if a new assignment is made before the clean up is done[8].

- **Affected files**:
  - `Sauron/CGIutil.pm`,
  - `Sauron/BackEnd.pm`
- **Patch links**:
  - http://tfm-mepl.googlecode.com/files/CGIutil.pm.patch (A.2)
  - http://tfm-mepl.googlecode.com/files/BackEnd.pm.patch (A.3)

Patches listed in C.2 and C.3 must be applied to fix this bug. These patches are also available to download from the above links. To apply it, download it and proceed as follows:

```
# cd /usr/local/sauron/
# patch -p1 -i /tmp/BackEnd.pm.patch
# patch -p1 -i /tmp/CGIutil.pm.patch
```

Note: You can simulate patching first by using `--dry-run` option. Remember to back up your files before to proceed.

[7] http://lists.jyu.fi/pipermail/sauron-users/2010/thread.html
[8] The assumption about the origin of the problem is based according to the mail list discussion.

A.3.1.3 PTR records: uppercase hostnames

An important problem when importing data is regarding to hostnames in uppercase. DNS is case insensitive for the names of the domain. Sauron import script is converting to lowercase all imported host names. In the process of importing, reverse zones are processed first in order to know whether create PTR records when direct zones are processed. But at this stage PTR records are parsed without perform lowercase conversion of host names. As a result, uppercase hosts defined in PTR records won't match their corresponding A records. To fix this bug, you can apply the following patch.

- **Affected files**:
  - `Sauron/UtilZone.pm`,
- **Patch links**:
  - `http://tfm-mepl.googlecode.com/files/UtilZone.pm.patch` (A.4)

Patch listed in C.4 must be applied to fix this bug. These patches are also available to download from the above links. To apply it, download it and proceed as follows:

```
# cd /usr/local/sauron/
# patch -p1 -i /tmp/UtilZone.pm.patch
```

Note: You can simulate patching first by using `--dry-run` option. Remember to back up your files before to proceed.

A.3.2 Feature improvement

A.3.2.1 IP–mask based permissions

Sauron comes with an option to apply constraints on user/groups on IP range basis. When a constraint is applied to some user, this one only will be able to add or modify hosts that match one of his constraints.

This feature should be fixed, since actually when an IP-mask constraint is declared, it is only being applied to host creation and not when editing an existing host. So if you define a IP-mask constraint for a user, this user will be able to create new hosts for the given IP range but will be able to modify other existing hosts out of given range too. To avoid this behaviour and apply a full-constraint on given IP range you can apply this patch.

- **Affected files**:
  - `Sauron/CGI/Hosts.pm`,
  - `Sauron/CGI/Utils.pm`
  - `moduser`

- **Patch links**:
  - * http://tfm-mepl.googlecode.com/files/Hosts.pm.restrict.patch (A.5)
  - * http://tfm-mepl.googlecode.com/files/Utils.pm.patch (A.6)
  - * http://tfm-mepl.googlecode.com/files/moduser.help.patch (A.7)

Patches listed in C.5, C.6 and C.7 must be applied to fix this feature. These patches are also available to download from the above links. To apply it, download it and proceed as follows:

```
# cd /usr/local/sauron/
# patch -p1 -i /tmp/Hosts.pm.restrict.patch
# patch -p1 -i /tmp/Utils.pm.patch
# patch -p1 -i /tmp/moduser.help.patch
```

Note: You can simulate patching first by using `--dry-run` option. Remember to back up your files before to proceed.

A.3.2.2 Disable HINFO option

This patch adds a new option in Sauron config file to completely disable HINFO fields in web interface. If you apply it, you will be able to decide if HINFO fields are shown or not in host records and add host forms.

The main reason for this is to decide whether HINFO records are required or not, because when enabled, this fields are mandatory for users with RW permission. Setting the option `$SAURON_DISABLE_HINFO = 1` in config file will disable HINFO for all users. If you are not making use of HINFO records this option may be useful.

- **Affected files**:
  - * `/usr/local/etc/sauron/config`,
  - * `Sauron/Sauron.pm`
  - * `Sauron/CGI/Hosts.pm`
- **Patch links**:
  - * http://tfm-mepl.googlecode.com/files/config.hinfo.patch (A.8)
  - * http://tfm-mepl.googlecode.com/files/Sauron.pm.hinfo.patch (A.9)
  - * http://tfm-mepl.googlecode.com/files/Hosts.pm.hinfo.patch (A.10)

Patches listed in C.8, C.9 and C.10 must be applied to add this feature. These patches are also available to download from the above links. To apply it, download it and proceed as follows:

```
# cd /usr/local/etc/sauron/
# patch -p1 -i /tmp/config.patch
# cd /usr/local/sauron/
# patch -p1 -i /tmp/Sauron.pm.hinfo.patch
# patch -p1 -i /tmp/Hosts.pm.hinfo.patch
```

Note: You can simulate patching first by using `--dry-run` option. Remember to back up your files before to proceed.

A.3.2.3 Import named.conf (include support)

Import script (`import`) does not support the include directive in `named.conf` file to include zones defined in separate files. So when you import BIND data, you need to replace all inclusions by the contents of included files.

The code to read file has been refactored in a subroutine to allow recursive call when an include is found.

Application of this patch adds support to include directive, but the path of included files is omited and *they are only read if are placed in the same directory where* `named.conf` *is read from*. This is to avoid problems with the paths and bind-chroot environments.

- **Affected files**:
  - `import`
- **Patch links**:
  - http://tfm-mepl.googlecode.com/files/import.patch (A.11)

Patch listed in C.11 must be applied to add this feature. This patch is also available to download from the above link. To apply it, download it and proceed as follows:

```
# cd /usr/local/sauron/
# patch -p1 -i /tmp/import.patch
```

Note: You can simulate patching first by using `--dry-run` option. Remember to back up your files before to proceed.

A.3.2.4 Import dhcpd.conf (include support)

As the avobe, `import-dhcp` script does not support include directive to include host declarations or any other data that you can have in separate files and you need to replace included files by its contents. But the path of included files is omited and *they are only read if are placed in the same directory where* `dhcpd.conf` *is read from*.

Applying this patch, include directive is supported and included files are processed as if their contents where at the point of include directive.

- **Affected files**:
  - `import-dhcp`
  - `Sauron/UtilDhcp.pm`
- **Patch links**:
  - * http://tfm-mepl.googlecode.com/files/import-dhcp.patch (A.12)
  - * http://tfm-mepl.googlecode.com/files/UtilDhcp.pm.patch (A.13)

Patches listed in C.12 and C.13 must be applied to add this feature. These patches are also available to download from the above links. To apply it, download it and proceed as follows:

```
# cd /usr/local/sauron/
# patch -p1 -i /tmp/import-dhcp.patch
# patch -p1 -i /tmp/UtilDhcp.pm.patch
```

Note: You can simulate patching first by using `--dry-run` option. Remember to back up your files before to proceed.

A.3.2.5 LDAP zone storage backend

The following patch is intended to provide support to LDAP backend for zone storage to files generated by Sauron for ISC BIND server. Usually BIND stores zone data in text files and loads its contents in memory at startup, but some database backends are available through `bind-sdb` package. This package provides some drivers that allows to store zone data in different kinds of databases including LDAP.

This patch adds an additional option to the script in charge to generate BIND configuration files. Specifying the option `--ldap=<URI>`, `sauron` script will generate suitable `named.conf/named.zones` to load data from a directory. An example of LDAP URI could be like this: `"ldap://ou=internal,ou=dns,dc=example,dc=com"`.

- **Affected file**:
  - `sauron`
- **Patch link**:
  - * http://tfm-mepl.googlecode.com/files/sauron.ldap.patch (A.14)

Patch listed in C.14 must be applied to add this feature. This patch is also available to download from the above link. To apply it, download it and proceed as follows:

```
# cd /usr/local/sauron/
# patch -p1 -i /tmp/sauron.ldap.patch
```

Note: You can simulate patching first by using `--dry-run` option. Remember to back up your files before to proceed.

A.4 Apache setup

In order to access to Sauron web interface, `sauron.cgi` and `browser.cgi` must be available through Apache web server. To do that you can copy or just make symbolic links to these files to Apache `cgi-bin` directory. Note that if you choose to create symbolic links, your Apache configuration must include the option `FollowSymLinks` as shown below for `cgi-bin` directory:

```
# cd /var/www/cgi-bin/
# ln -s /usr/local/sauron/cgi/sauron.cgi
# ln -s /usr/local/sauron/cgi/browser.cgi
```

```
<Directory "/var/www/cgi-bin">
    AllowOverride None
    Options FollowSymLinks
    Order allow,deny
    Allow from all
</Directory>
```

file: /etc/httpd/conf/httpd.conf

Now you need to create the log file for web interface and set Apache as owner. In CentOS, Apache user is `apache`, but note that this name can be different in other distributions (e.g. www-data,...). Check the name in `httpd.conf` and proceed as follows:

```
# cd /usr/local/sauron/
# touch logs/sauron.log
# chown apache:apache logs/sauron.log
```

Finally you need to start Apache service and set it up to auto-start when you boot your system:

```
# chkconfig httpd on
# service httpd start
```

At this point Sauron interface is ready to operate. You should be able to open it in your browser through `http://<SAURON_IP>/cgi-bin/sauron.cgi` and login with the administrator user you created before.

A.5 Importing your DNS and DHCP data

Sauron works generating the configuration and data files for BIND DNS server and ISC DHCP. You can start adding a new server from scratch, but Sauron also comes with a set of scripts to import the current data and configuration of your servers. Once done, you will be able to manage the data and tweak configuration from Sauron interface.

If you have applied the patch listed in C.1 as described in section A.3, you should be able to import the data from your production servers. To do that a couple of scripts are provided:

- `import`: import BIND configuration and data to Sauron.
- `import-dhcp`: add information from DHCP to Sauron.

Refer to *Sauron User Guide* for further information or execute it with `--help` option.

Some things to notice

(i) Each configuration and particular case can end up with unexpected behavior, so ensure to review the log provided by import scripts to know if something was wrong or omitted. You can make use of `--verbose` option to increase output logging.

(ii) Import scripts (`import` and `import-dhcp`) by `named.conf` and `dhcpd.conf` files does not support `include` directives in order to read data provided in separate files. So you need to replace includes by the contents of included files or you can apply the patches described in A.3.2.3 listed in C.11 to add support for this feature.

(iii) Another thing to notice is the location of your BIND files. If you are importing data from a bind-chroot environment, you should place your files to the directory specified in `named.conf` or use the `--dir` option in import script.

(iv) An important problem when importing data is regarding to hostnames in uppercase. DNS is case insensitive for the names of the domain. Sauron import script is converting to lowercase all imported host names. In the process of importing, reverse zones are processed first in order to know whether create PTR records when direct zones are processed. But at this stage PTR records are parsed without perform lowercase conversion of host names. As a result, uppercase hosts defined in PTR records won't match their corresponding A records. To fix this bug, you can apply the patch A.3.1.3, listed in C.4.

A.6 Working around your network

Once you have imported all your network data —this is BIND and DHCP data— you can start customizing some things such as DNS global options that has been skipped when importing, network topology and so on.

You can configure your network topology: Nets, Subnets and VLANs. A proper configuration of the topology, will let you two things:

(i) auto-generate updated DHCP configuration (`dhcpd.conf`),

(ii) manage your hosts (add, edit, move,...) according with the defined subnets.

Subnets and VLANs (shared-networks) are created when importing DHCP data from `dhcpd.conf`, but to generate a valid DHCP configuration, manually creation of a parent net for each group of subnets is required. This parent net only has to be a subnet holder and you can keep defining global network options in VLAN section. This is necessary because Sauron will only create DHCP entries for those subnets that fit in any of declared networks, but the later ones are not created automatically.

If your organization is not actually using subneting (i.e. you have a single large subnet where all departments are addressed and communicate without routing) but you want to keep a logical division of these departments based on address ranges, you can define as subnet (increasing the mask) each range of addresses that have to be DHCP managed. Remember to create at least one subnet that holds each address where your DHCP server is listening or your server won't serve at this interface.

But take care with DHCP! If you create subnets, original netmask will be modified. You should define `subnet-mask` and `broadcast-address` options according to your network. A good idea is to define a VLAN, assign subnets to that VLAN and declare all common options including the above ones there.

Doing this way, Sauron will let you to work around your single subnet network but with an address range division (e.g. add new hosts to specified IP range and assign it the first free address,...).

A.7 Adding users, groups and permissions

Seuron let you to create users and groups of users, grant some privileges and apply some constraints to them. This will allow you to delegate part of the management of a zone to other zone administrators than superuser.

You can grant (R, RW or RWX[9]) permissions to users over a server and zone. Typically you will want to delegate the management of a zone. This can be achieved by granting (R) read to server and (RW) write to zone.

If you don't want to grant permissions over whole hosts in a zone, you can define some constraints such as CIDR net, hostname mask, IP mask or a combination of them.

Finer grain permission control can be achieved by using IP-mask but patches described in A.3.2.1, listed in C need to be applied. This allow you to grant permissions on a specified IP range with no need to fit CIDR address. You can specify ranges in this format: e.g. `192.168.1-5.*`

User/group management need to be done through command-line interface. `addgroup`, `adduser`, `modgroup` and `moduser` scripts are provided and available under Sauron installation directory. You can run it with `--help` option or refer to *Sauron User Guide* for further details about the usage.

Tip: To list the constraints defined by user or group, run `./modgroup <groupname>`.

A.8 Generating BIND and DHCP files

Once all your network data is imported to Sauron, configuration files for BIND and DHCP can be generated and checked automatically. The command `./status --pending` will inform if there are pending changes since the last time configuration files was generated. If so, you can launch sauron again to get an updated version of these configurations.

If you applied the patch described in A.3.2.5, listed in C.14, Sauron will be able to generate suitable `named.conf/named.zones` to read zone data from an LDAP directory. You can run `./sauron --help` to see available options when generating files.

```
# ./sauron --verbose --check --bind <SERVER_NAME> /tmp/sauron-bind/
```

will generate updated BIND zone files and `named.conf` or `named.zones`, depending on your server preferences.

```
# ./sauron --verbose --check --dhcp <SERVER_NAME> /tmp/sauron-dhcp/
```

will do the same for DHCP.

The best way to do this is to write a bash script scheduled with cron and check for the status. Only if they are pending changes, generate configurations and move it to their corresponding targets.

[9] RWX is a less restrictive Read/Write permission. Refer to *Sauron User Guide* for further details of RWX.

Note: You can uncomment `$SAURON_DHCP_CHK_PROG` and `$SAURON_NAMED_CHK_PROG` options in config file in order to check generated files.

Appendix B

BIND dynamic backend and OpenLDAP: Setup and deployment

This appendix is devoted to the setup and deployment of a ISC BIND DNS server using an LDAP directory as storage backend. That means that zone data that typically BIND reads from text files and load in memory at start-up is in this case stored in a database and accessed as requested. This configuration brings some benefits such as dynamic data loading, no need to reload service on changes, no need of slave servers neither zone transfers, and getting rid about cryptic zone files, among others.

These pages will guide you through the setup of BIND and OpenLDAP to get the system working in a production environment making use of OpenLDAP replication to set up several authoritative primary name severs for your zones. More details about architecture are shown in the following sections.

Note: *This appendix is based in GNU/Linux CentOS 6.0.*

B.1 Overview

BIND is shipped with a driver that allows to store and read zone data from an LDAP server, provided by `bind-sdb` package available —at least— at RHEL/CentOS Linux distributions. The overall idea is to feed a directory with the zone files generated by Sauron and make BIND to read from this directory. To feed the LDAP with zone files you can download and use `syncldapzone.pl` script from the link below:

* `http://tfm-mepl.googlecode.com/files/syncldapzone.pl` (B.1)

This Perl program reads zone files and synchronize data with LDAP directory, mirroring a local zone file to directory. When a RR is modified in local zone file, only the

corresponding LDAP entry is updated. Using it together with Sauron it gives you a good consistency schema.

Note: *This script has not been extensively tested. I wrote it to get in sync my Sauron-generated zone files with LDAP and, as I could test, it works fine. You're free to copy, modify or improve it as you need.*

It is recommended to patch Sauron as described in A.3.2.5 in order to generate suitable `named.conf/named.zones` file to use LDAP backend.

B.1.1 System arquitecture

As already explained, the default in-memory storage of bind will be replaced by a database backend (an LDAP directory). To add reliability to the whole domain name system, setting up at least two authoritative name servers for each zone is required. Keeping in mind that in this setup name servers are reading data from a directory, an LDAP server for each DNS server must be set up. The best way is that each name server runs his own LDAP server working as a *consumer* of a master LDAP *provider*. These consumers are working as read-only directories and are updated from a master directory (the provider) using the OpenLDAP *syncrepl* protocol. Modifications are only written down to *provider* and with a properly configured replication, every change is immediately pushed to each consumer.

This configuration eliminates the single point of failure that would suppose the use of a central directory, minimises the network traffic and increases performance, maximising the availability of the system. Figure B.1 shows a diagram of the system architecture.

As you can see from the figure, *provider* database is split in two parts: internal and external. This represents two views of DNS data and can be achieved by storing each view in a separate sub-tree (e.g. "ou=internal,ou=dns,dc=example,dc=com" and "ou=external,ou=dns,dc=example,dc=com"). The way to deal with views in Sauron is by adding a server for each view and importing the zones from the view to each server. So it would be possible to replicate the whole database and use BIND views to control data access.

The communication between consumers and provider can be performed with TLS using public certificates, which gives an strong authentication mechanism and data integrity, providing protection against eavesdropping, poisoning or MitM attacks. Consumers, who start the connection, can authenticate provider in a secure way.

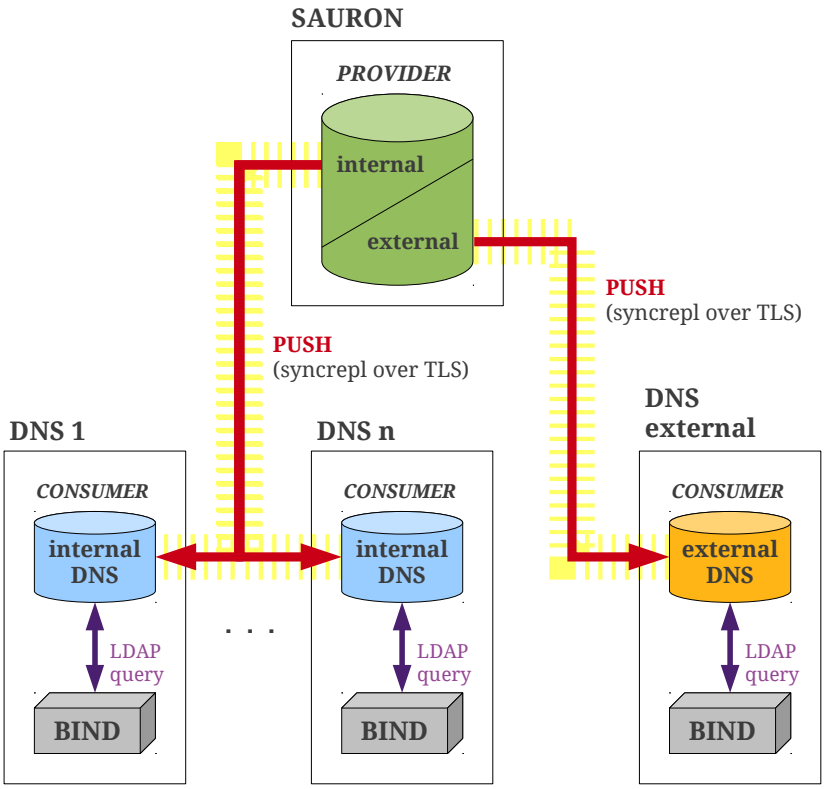

Figure B.1: BIND and LDAP system architecture

B.2 BIND setup

To work with LDAP backend you need to install `bind-sdb` package available from CentOS repositories via **yum**. No special configuration is required further than tell `named.conf` to read data from database instead of file in zone declaration.

If you are using Sauron patched version, generated `named.conf/named.zones` is already done. If not, you only need to change —for each zone to be loaded from LDAP— the line:

```
file "example.com.zone";
```

for,

```
database "ldap ldap://127.0.0.1/ou=internal,ou=dns,dc=foo,dc=org 86400";
```

or whatever host, DN and default TTL you want to read data from —even though read from local directory is recommended as explained in previous section. It should be noticed than version 1.0 of bind-sdb does not support neither `ldaps` nor `ldapi` URI schemas, so use `ldap://`. For more details visit http://bind9-ldap.bayour.com/.

Important: As named service (BIND) is depending on slapd service (OpenLDAP) you need to modify the startup scripts priority to start slapd before named. You can do it by modifying the line,

```
# chkconfig:  - SS KK
```

from `/etc/init.d/{named, slapd}` files, where SS and KK represents the order in booting sequence (Start) and shutdown (Kill) of services. Change SS values, do not uncomment line. Then run,

```
# chkconfig slapd off; chkconfig slapd on
# chkconfig named off; chkconfig named on
```

to generate symbolic links with updated sequence numbers. Just make sure that slapd starts before named or BIND won't load zones at boot time.

B.3 OpenLDAP setup

Setting up OpenLDAP is quiet more complex than BIND, so we are going to put more attention at this section.

Since version 2.3, OpenLDAP comes with a run-time configuration backend (RTC): *cn=config*. This new backend is intended to replace static slapd.conf configuration and allows that changes in configuration can be applied without restarting the service. This configuration is stored in the same DIT and can be managed using ldap client tools on-the-fly. Even though static slapd.conf is still supported, it is recommended to use the new *cn=config* (see `man slapd-config` for available options and refer to *OpenLDAP Admin Guide*[1] for further details). Figure B.2 shows a diagram of RTC DIT. Configuration of LDAP and databases is located below `/etc/openldap/slap.d/` in several LDIF files. This files can be dynamically updated using ldap client tools.

Note: This section is based in OpenLDAP 2.4.

B.3.1 Installation and basic setup

First of all you need to install `openldap-servers` and `openldap-clients` packages using **yum**. After that you should configure rsyslog facility to handle LDAP log. Add the following line to `/etc/rsyslog.conf`,

```
local4.*                -/var/log/ldap.log
```

Leading dash "-" indicates to not sync file immediately after every logging, which can improve performance.

[1] http://www.openldap.org/doc/admin24/

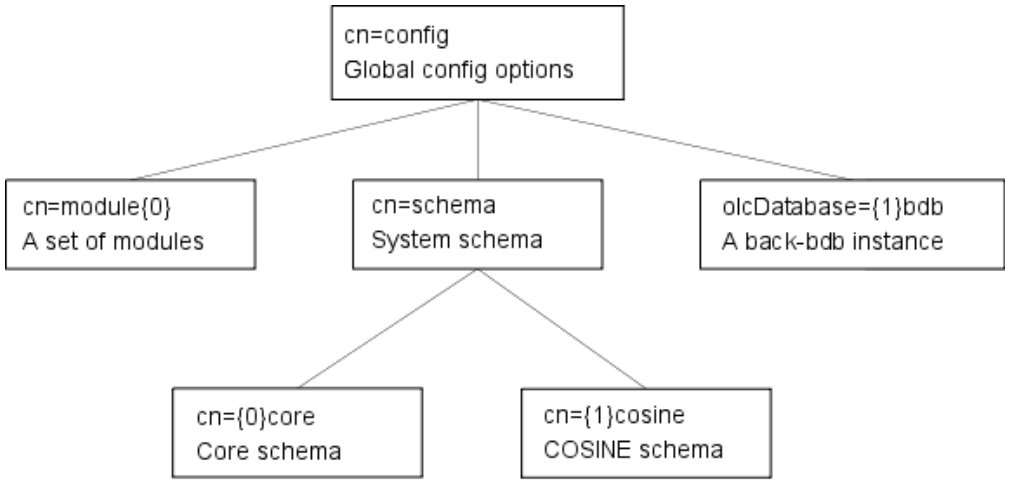

Figure B.2: Run-time configuration DIT

Restart rsyslog service:

```
# service rsyslog restart
```

Now you have to edit the database suffix, according to your domain name, in config files:

```
# cd /etc/openldap/slapd.d/cn\=config/
# sed -i -e 's/dc=my-domain,dc=com/dc=example,dc=com/g' olcDatabase\=\{1\}bdb.ldif
# sed -i -e 's/cn=manager,dc=my-domain,dc=com/cn=Manager,dc=example,dc=com/g'
olcDatabase\=\{2\}monitor.ldif
```

You need to generate passwords for config admin and database Manager. Use `slappasswd` for that purpose which generates SHA-1 hashed passwords from a passphrase. Then you must add these generated password-hashes using `olcRootPW` directive to corresponding LDIF files as shown:

```
olcRootDN: cn=Manager,dc=udl,dc=cat
olcRootPW: {SSHA}JjbbbsiWyLgaZifjAX9v1O1MOMEZn8ZX
```

file: /etc/openldap/slapd.d/cn=config/olcDatabase={1}bdb.ldif

```
olcRootDN: cn=admin,cn=config
olcRootPW: {SSHA}uVhLAlpbd39JHdnU4kUagRaEeQ5Fs+0e
```

file: /etc/openldap/slapd.d/cn=config/olcDatabase={0}config.ldif

Then copy this file:

```
# cp /usr/share/doc/openldap-servers-2.4.19/DB_CONFIG.example
/var/lib/ldap/DB_CONFIG
# chown -Rf ldap:ldap /var/lib/ldap/
# updatedb
```

This is your database directory and all files under this directory must be owned by ldap. Finally, modify `/etc/sysconfig/ldap` to allow TLS connections:

```
SLAPD_LDAPS=yes
```

You can test your configuration using `slaptest -u` and start service:

```
# service slapd start
```

Note: *Remember to modify service boot priority as explained in previous section to make slapd start before named.*

Adding base domain

If LDAP starts fine you can add the base domain. Listing D.1 shows an example of DIT you can use to store your DNS zones. You have to load it to directory using `ldapadd`:

```
# ldapadd -x -D cn=Manager,dc=example,dc=com -W -f dnsbase.ldif
```

B.3.2 dNSZone schema

To store DNS data, an additional schema must be installed to LDAP. dNSZone schema defines some of the attributes and object classes required. This schema extends some of the attributes defined in cosine schema (shipped with OpenLDAP); so both are necessary. dNSZone is provided by bind-sdb package but to use it with cn=config backend it must be converted to LDIF. You can download converted version, listed at D.7 from the following link:

* http://tfm-mepl.googlecode.com/files/dnszone.ldif (B.2)

Then, add it to ldap using `ldapadd` as usual:

```
# ldapadd -x -D cn=admin,cn=config -W -f dnszone.ldif
```

And now add indices to your database to improve search performance. Use LDIF file listed in D.2:

```
# ldapadd -x -D cn=admin,cn=config -W -f dnsindices.ldif
```

B.3.3 Setting up certificates

To allow consumers authenticate provider (and talk over TLS), setting up at least a certificate for provider is required.

If you don't have access to a CA to sign your certificate, you can create self-signed certificate using OpenSSL as shown below and use it then as a CA certificate for consumers:

```
# openssl req -new -x509 -nodes -out /etc/pki/tls/certs/slapd-cert.pem
-keyout /etc/pki/tls/certs/slapd-key.pem -days 999999
```

Important note!

To make authentication working, the CN attribute of the certificate must be the be the server's fully qualified domain name (FQDN) —without trailing dot.

and set the permissions and owner:

```
# chown root:ldap /etc/pki/tls/certs/slapd-cert.pem
# chown ldap:ldap /etc/pki/tls/certs/slapd-key.pem
# chmod 400 /etc/pki/tls/certs/slapd-key.pem
```

Once certificates are generated is the time to set it up to OpenLDAP configuration. To do that we can make use of RTC and using `ldapadd`, load the required directives to the configuration. You can download from the following link the LDIF file listed in D.3 and and use it to feed LDAP:

* http://tfm-mepl.googlecode.com/files/tlsconfig.ldif (B.3)

```
# ldapadd -x -D cn=admin,cn=config -W -f tlsconfig.ldif
```

Consumer (LDAP slave)

In replication mechanism, each consumer establishes a connection with provider at start up and start syncing data. So each consumer, in order to authenticate provider, need to set up the CA certificate to verify provider's certificate authenticity. This configuration is made from the client side of consumer since when replicates, consumer acts as provider's client. You need to specify the path to CA cert at `ldap.conf`. To do it just add the following lines,

```
TLS_CACERT      /etc/pki/tls/certs/myCA.pem
TLS_REQCERT     demand
```

to `/etc/openldap/ldap.conf`, and copy your self-signed provider certificate (or your CA certificate) to specified location. Remember that if you generated a self-signed certificate on provider, that one is what will be the CA certificate for your consumers.

B.3.4 Replication

OpenLDAP 2.3 introduced *syncrepl* as a new replication mechanism. The new replication system provides more flexibility and reliability and has completely replaced the old *slurpd* since version 2.4.

In this scenario, each DNS server (the consumers) will replicate to their local directories the data from the provider directory. In order to reduce out-of-date data time to zero, replication will be configured to push immediately changes on provider to every connected consumer. OpenLDAP calls this mechanism *refreshAndPersist*. DNS directories will act as read-only directories, so modifications will be performed on master (or provider) directory.

Consumers are working as clients from provider database and the later doesn't has to have previous knowledge about consumers. This provides a powerful and very easy way to add consumers on-the-fly.

Also may be interesting to create a user at provider directory which will be used for consumers to replicate data. Steps to set up replication are shown below.

Provider side

You need to add a replicator user and load syncrepl module at provider. Replicator user will be configured with unlimited read access. Download LDIFs listed in D.4 and D.5 from,

* `http://tfm-mepl.googlecode.com/files/replicator-user.ldif` (B.4)

* `http://tfm-mepl.googlecode.com/files/provider.ldif` (B.5)

and add it with `ldapadd` in this order:

Note that you need to change `userPassword` from `replicator-user.ldif`. Use `ldappasswd` to generate your own.

```
# ldapadd -x -D cn=Manager,dc=example,dc=com -W -f replicator-user.ldif
# ldapadd -x -D cn=admin,cn=config -W -f provider.ldif
```

Consumer side

At consumer side you need to specify provider's parameters using `olcSyncRepl` directive. You can download the LDIF listed at D.6 and add it as usual.

* `http://tfm-mepl.googlecode.com/files/consumer.ldif` (B.6)

Notice that you need to modify the parameters of LDIF file according to your network and setup. Change `<NUM>`, the provider `hostname` and `<PASS>` to correspond your setup. `rid` parameter is the replicator identifier and is a 3 digit number that must be unique for each consumer.

```
ldapadd -x -D cn=admin,cn=config -W -f consumer.ldif
```

Important note!

*Consumer must specify the domain name of provider to reach them and **not** IP address, since provider certificate contains their FQDN, and authentication would not succeed using IP. But given that in this context consumer hosts are pieces of domain name system —actually they are the name servers— you may need to define at `/etc/hosts` the IP address and domain name of provider in order to reach them by their name.*

Appendix C

Patching sources: diff files

This appendice is intended to list the code of the patches regarding to the section A.3. The following listings shows the output of patches generated by *GNU diff* in unified format and can be applied using *GNU patch* tool.

Patch 1: DB-DBI.pm

Listing C.1: Sauron/DB-DBI.pm patch

```
--- a/Sauron/DB-DBI.pm  2011-08-07 00:32:08.133157526 +0200
+++ b/Sauron/DB-DBI.pm  2011-08-07 00:36:42.643361872 +0200
@@ -247,10 +247,9 @@

 sub db_insert($$$) {
   my($table,$fields,$data) = @_;
-  my($str,$i,$j,$c,$row,$flag,$res);
+  my($str,$i,$j,$row,$flag,$res);


-  $c=0;
   for $i (0..$#{$data}) {
     $row=$$data[$i]; $flag=0;
     $str.="INSERT INTO $table ($fields) VALUES(";
@@ -260,20 +259,10 @@
       $flag=1;
     }
     $str.=");\n";
-    $c++;
-    if ($c > 25) {
-      $c=0;
-      #print "BLOCK: $str\n";
-      $res=db_exec($str);
```

```
-       return -1 if ($res < 0);
-       $str='';
-     }
-   }
-
-   if ($str ne '') {
-     #print "LAST: $str\n";
+     #print "STATEMENT: $str\n";
      $res=db_exec($str);
-     return -2 if ($res < 0);
+     return -1 if ($res < 0);
+     $str='';
    }

    return 0;
```

Patch 2: CGIutil.pm

Listing C.2: Sauron/CGIutil.pm patch

```
--- a/Sauron/CGIutil.pm 2011-08-04 15:32:16.700878339 +0200
+++ b/Sauron/CGIutil.pm 2011-08-04 17:05:12.000000000 +0200
@@ -881,7 +881,11 @@
         print "</TABLE></TD>";
       }
       elsif ($rec->{ftype} == 101) {
-        undef @q; undef @lst; undef %lsth;
+        #undef @q; undef @lst; undef %lsth;
+        @q = [];
+        @lst = [];
+        $lst[0] = '';    # initialize the empty element
+        %lsth = {};
         $maxlen=$rec->{len};
         $maxlen=$rec->{maxlen} if ($rec->{maxlen} > 0);
         db_query($rec->{sql},\@q);
@@ -890,7 +894,7 @@
           $lsth{$q[$i][0]}=$q[$i][0];
         }
         if ($rec->{addempty} > 0) {
-          push @lst,'';
+          #push @lst,'';  # commented out 20100608 rtb
           $lsth{''}='<none>';
         }
         elsif ($rec->{addempty} < 0) {
```

Patch 3: BackEnd.pm

Listing C.3: Sauron/BackEnd.pm patch

```
--- a/Sauron/BackEnd.pm 2011-08-04 15:32:16.700878339 +0200
+++ b/Sauron/BackEnd.pm 2011-08-04 17:03:41.000000000 +0200
@@ -2051,7 +2051,7 @@

    undef @{$lst};
    push @{$lst}, -1;
-   undef %{$rec};
+   #undef %{$rec};
    $$rec{-1}='--None--';
    return if ($zoneid < 1);
    $alevel=0 unless ($alevel>0);
@@ -2150,7 +2150,7 @@

    undef @{$lst};
    push @{$lst}, -1;
-   undef %{$rec};
+   #undef %{$rec};
    $$rec{-1}='--None--';
    return if ($serverid < 1);
    $alevel=0 unless ($alevel > 0);
@@ -2397,7 +2397,7 @@

    undef @{$lst};
    push @{$lst}, -1;
-   undef %{$rec};
+   #undef %{$rec};
    $$rec{-1}='--None--';
    return unless ($serverid > 0);
    $alevel=0 unless ($alevel > 0);
```

Patch 4: PTR records, uppercase hostnames

Listing C.4: Sauron/UtilZone.pm patch

```
--- a/Sauron/UtilZone.pm        2011-08-07 00:32:08.133157526 +0200
+++ b/Sauron/UtilZone.pm        2011-08-22 19:23:36.000000000 +0200
@@ -208,7 +208,8 @@
        $rec->{SOA} = join(" ",@line);
      }
      elsif ($type eq 'PTR') {
-       push @{$rec->{PTR}}, $line[0];
+       #lowercase PTR in order to match, like host names do - gbosch 2011-08-22
+       push @{$rec->{PTR}}, "\L$line[0]";
      }
      elsif ($type eq 'CNAME') {
        $rec->{CNAME} = add_origin($line[0],$origin);
```

Patches 5, 6, 7: IP based permissions

Listing C.5: CGI/Hosts.pm patch

```
--- a/Sauron/CGI/Hosts.pm       2011-08-07 00:32:08.133157526 +0200
+++ b/Sauron/CGI/Hosts.pm       2011-08-26 17:15:06.920199024 +0200
@@ -508,7 +508,10 @@

   if ($sub eq 'Delete') {
     return unless ($id > 0);
+
     goto show_host_record if (check_perms('delhost',$host{domain}));
+    #added to restrict host deletion on IP basis - gbosch June 2010:
+    goto show_host_record if (check_perms('ip',$host{ip}[1][1]));

     $res=delete_magic('h','Host','hosts',\%host_form,\&get_host,\&delete_host,
                       $id);
@@ -523,7 +526,11 @@
   }
   elsif ($sub eq 'Disable') {
     return unless ($id > 0);
+
     goto show_host_record if (check_perms('delhost',$host{domain}));
+    #added to restrict host 'Disable' on IP basis permissions - gbosch 2010:
+    goto show_host_record if (check_perms('ip',$host{ip}[1][1]));
+
     if (update_host({id=>$id,type=>101}) < 0) {
       alert2("Failed to update host record (id=$id)");
     } else {
@@ -562,7 +569,10 @@
   }
   elsif ($sub eq 'Move') {
     return unless ($id > 0);
+
     goto show_host_record if (check_perms('host',$host{domain}));
+    #added to restrict host 'Move' on IP basis permissions - gbosch 2010:
+    goto show_host_record if (check_perms('ip', $host{ip}[1][1]));

     if ($#{$host{ip}} > 1) {
       alert2("Host has multiple IPs!");
@@ -707,7 +717,11 @@
   elsif ($sub eq 'Edit' || $sub eq 'Enable') {
     $host{type}=1 if ($sub eq 'Enable');
     return unless ($id > 0);
+
     goto show_host_record if (check_perms('host',$host{domain}));
+    #added to restrict host edition on IP basis - gbosch June 2010:
+    goto show_host_record if (check_perms('ip',$host{ip}[1][1]));
+
     my $hform=(check_perms('zone','RWX',1) ?
                \%restricted_host_form :\%host_form);
```

Listing C.6: CGI/Utils.pm patch

```
--- a/Sauron/CGI/Utils.pm       2011-08-07 00:32:08.133157526 +0200
+++ b/Sauron/CGI/Utils.pm       2011-08-09 21:33:59.921836257 +0200
@@ -123,7 +123,7 @@
         return 0 if (check_ipmask($re,$rule));
     }

-    alert1("Invalid IP (IP is outsize allowed net(s))") unless ($quiet);
+    alert1("You are not authorized to modify this host") unless ($quiet);
     return 1;
   }
   elsif ($type eq 'tmplmask') {
```

Listing C.7: moduser patch

```
--- a/moduser   2011-08-07 00:32:08.133157526 +0200
+++ b/moduser   2011-08-10 10:31:16.749473605 +0200
@@ -213,7 +213,7 @@
         "   --add=\"hostmask,<regex>\"                add global hostname mask\n",
         "   --add=\"hostmask,<servername>:<zonename>,<regex>\"\n",
         "                                          add zone-bound hostname mask\n",
-        "   --add=\"ipmask,<ipmask>\"                  add IP-mask\n",
+        "   --add=\"ipmask,<ipmask>\"                  add IP-mask (e.g. 192.168.0-5.*)\n",
         "   --add=\"level,<level>\"                     add priv.level\n",
         "   --add=\"elimit,<days>\"                     add expiration limit\n",
         "   --add=\"def_dept,<string>\"                 add default dept for hosts\n",
```

Patches 8, 9, 10: Disable HINFO option

Listing C.8: etc/sauron/config patch

```
--- a/config    2011-08-10 10:13:25.673454514 +0200
+++ b/config    2011-08-10 10:19:55.669484799 +0200
@@ -181,6 +181,13 @@
 # set 1 to use hardcoded DTD instead the one genereated by CGI module
 # $SAURON_DTD_HACK = 0;

+# gbosch 2011-08-08
+# Enable/Disable HINFO fields in Host form. If disabled, HINFO fields
+# are not displayed, no value required neither HINFO records generated.
+# 1=disabled | 0=enabled (default)
+#
+# $SAURON_DISABLE_HINFO = 1;
+
 # Set this to change behaviour of HINFO edit fields (for hosts):
 #    1 = Allow empty values (last in the drop-down list)
 #   -1 = Allow emptry values (first in the drop-down list)
```

Listing C.9: Sauron/Sauron.pm patch

```
--- a/Sauron/Sauron.pm  2011-08-10 01:43:22.244577338 +0200
+++ b/Sauron/Sauron.pm  2011-08-10 01:06:45.000000000 +0200
@@ -74,6 +74,7 @@
    $main::SAURON_ZONE_CHK_PROG = '';
    $main::SAURON_ZONE_CHK_ARGS = '-q';
    $main::SAURON_NO_REMOTE_ADDR_AUTH = 0;
+   $main::SAURON_DISABLE_HINFO = 0;
    $main::SAURON_HINFO_MODE = 1;
    $main::SAURON_PLUGINS = '';
    $main::SAURON_KEY = '';
@@ -150,6 +151,7 @@
    print "SAURON_DNSSEC_KEYGEN_PROG=",$main::SAURON_DNSSEC_KEYGEN_PROG,"\n";
    print "SAURON_DNSSEC_KEYGEN_ARGS=",$main::SAURON_DNSSEC_KEYGEN_ARGS,"\n";
    print "SAURON_NO_REMOTE_ADDR_AUTH=",$main::SAURON_NO_REMOTE_ADDR_AUTH,"\n";
+   print "SAURON_DISABLE_HINFO=",$main::SAURON_DISABLE_HINFO,"\n";
    print "SAURON_HINFO_MODE=",$main::SAURON_HINFO_MODE,"\n";
    print "SAURON_DNSNAME_CHECK_MODE=",$main::SAURON_DNSNAME_CHECK_MODE,"\n";
    print "SAURON_PLUGINS='",$main::SAURON_PLUGINS,"'\n";
```

Listing C.10: CGI/Hosts.pm patch

```
--- a/Sauron/CGI/Hosts.pm       2011-08-07 00:32:08.133157526 +0200
+++ b/Sauron/CGI/Hosts.pm       2011-08-09 02:35:27.191952807 +0200
@@ -26,6 +26,55 @@
                        $main::SAURON_HINFO_MODE : 1);
 my $chr_group;

+
+# HINFO fields in host form
+my @hinfo_hf = (
+  {ftype=>101, tag=>'hinfo_hw', name=>'HINFO hardware', type=>'hinfo', len=>25,
+   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=0 ORDER BY pri,hinfo;",
+   addempty=>$hinfo_addempty_mode, empty=>1, iff=>['type','[19]']},
+
+  {ftype=>101, tag=>'hinfo_sw', name=>'HINFO software', type=>'hinfo',len=>25,
+   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=1 ORDER BY pri,hinfo;",
+   addempty=>$hinfo_addempty_mode, empty=>1, iff=>['type','[19]']}
+);
+
+# HINFO fields in restricted host form
+my @hinfo_rhf = (
+  {ftype=>101, tag=>'hinfo_hw', name=>'HINFO hardware', type=>'hinfo', len=>25,
+   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=0 ORDER BY pri,hinfo;",
+   addempty=>$hinfo_addempty_mode, empty=>$main::SAURON_DISABLE_HINFO,
+   iff=>['type','[19]']},
+
+   {ftype=>101, tag=>'hinfo_sw', name=>'HINFO software', type=>'hinfo',len=>25,
+   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=1 ORDER BY pri,hinfo;",
+   addempty=>$hinfo_addempty_mode, empty=>$main::SAURON_DISABLE_HINFO,
+   iff=>['type','[19]']}
+);
+
+# HINFO fields in new host form
+my @hinfo_nhf = (
+  {ftype=>101, tag=>'hinfo_hw', name=>'HINFO hardware', type=>'hinfo', len=>20,
+   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=0 ORDER BY pri,hinfo;",
+   addempty=>$hinfo_addempty_mode, empty=>1, iff=>['type','1']},
```

```
+
+{ftype=>101, tag=>'hinfo_sw', name=>'HINFO software', type=>'hinfo',len=>20,
+   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=1 ORDER BY pri,hinfo;",
+   addempty=>$hinfo_addempty_mode, empty=>1, iff=>['type','1']}
+);
+
+# HINFO fields in new-restricted host form
+my @hinfo_nrhf = (
+  {ftype=>101, tag=>'hinfo_hw', name=>'HINFO hardware', type=>'hinfo', len=>20,
+   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=0 ORDER BY pri,hinfo;",
+   addempty=>$hinfo_addempty_mode, empty=>$main::SAURON_DISABLE_HINFO,
+   iff=>['type','1']},
+
+  {ftype=>101, tag=>'hinfo_sw', name=>'HINFO software', type=>'hinfo',len=>20,
+   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=1 ORDER BY pri,hinfo;",
+   addempty=>$hinfo_addempty_mode, empty=>$main::SAURON_DISABLE_HINFO, iff=>['type','1']}
+);
+
+
 my %host_form = (
   data=>[
   {ftype=>0, name=>'Host' },
@@ -62,12 +111,7 @@
    maxlen=>100, empty=>1},

   {ftype=>0, name=>'Equipment info', iff=>['type','1|9|101']},
-  {ftype=>101, tag=>'hinfo_hw', name=>'HINFO hardware', type=>'hinfo', len=>25,
-   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=0 ORDER BY pri,hinfo;",
-   addempty=>$hinfo_addempty_mode, empty=>1, iff=>['type','[19]']},
-  {ftype=>101, tag=>'hinfo_sw', name=>'HINFO software', type=>'hinfo',len=>25,
-   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=1 ORDER BY pri,hinfo;",
-   addempty=>$hinfo_addempty_mode, empty=>1, iff=>['type','[19]']},
+  #HINFO appended here if enabled (pos 20)
   {ftype=>1, tag=>'ether', name=>'Ethernet address', type=>'mac', len=>17,
    conv=>'U', iff=>['type','([19]|101)'], empty=>1},
   {ftype=>4, tag=>'card_info', name=>'Card manufacturer',
@@ -156,12 +200,7 @@
    maxlen=>100, empty=>$main::SAURON_RHF{info}},

   {ftype=>0, name=>'Equipment info', iff=>['type','1|9|101']},
-  {ftype=>101, tag=>'hinfo_hw', name=>'HINFO hardware', type=>'hinfo', len=>25,
-   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=0 ORDER BY pri,hinfo;",
-   addempty=>$hinfo_addempty_mode, empty=>1, iff=>['type','[19]']},
-  {ftype=>101, tag=>'hinfo_sw', name=>'HINFO software', type=>'hinfo',len=>25,
-   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=1 ORDER BY pri,hinfo;",
-   addempty=>$hinfo_addempty_mode, empty=>1, iff=>['type','[19]']},
+  #HINFO appended here if enabled (pos 12)
   {ftype=>1, tag=>'ether', name=>'Ethernet address', type=>'mac', len=>17,
    conv=>'U', iff=>['type','[19]'], iff2=>['ether_alias_info',''],
    empty=>$main::SAURON_RHF{ether}},
@@ -247,12 +286,7 @@
   {ftype=>1, tag=>'info', name=>'Info', type=>'text', len=>50, maxlen=>100,
    empty=>1 },
   {ftype=>0, name=>'Equipment info',iff=>['type','1']},
-  {ftype=>101, tag=>'hinfo_hw', name=>'HINFO hardware', type=>'hinfo', len=>20,
-   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=0 ORDER BY pri,hinfo;",
-   addempty=>$hinfo_addempty_mode, empty=>1, iff=>['type','1']},
-  {ftype=>101, tag=>'hinfo_sw', name=>'HINFO software', type=>'hinfo',len=>20,
-   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=1 ORDER BY pri,hinfo;",
-   addempty=>$hinfo_addempty_mode, empty=>1, iff=>['type','1']},
+  #HINFO appended here if enabled (pos 27)
   {ftype=>1, tag=>'ether', name=>'Ethernet address', type=>'mac', len=>17,
```

```
    conv=>'U', iff=>['type','(1|9|101)'], empty=>1},

@@ -321,12 +355,7 @@
   {ftype=>1, tag=>'info', name=>'[Extra] Info', type=>'text', len=>50,
    maxlen=>100, empty=>$main::SAURON_RHF{info} },
   {ftype=>0, name=>'Equipment info', iff=>['type','1']},
-  {ftype=>101, tag=>'hinfo_hw', name=>'HINFO hardware', type=>'hinfo', len=>20,
-   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=0 ORDER BY pri,hinfo;",
-   addempty=>$hinfo_addempty_mode, empty=>0, iff=>['type','1']},
-  {ftype=>101, tag=>'hinfo_sw', name=>'HINFO software', type=>'hinfo',len=>20,
-   sql=>"SELECT hinfo FROM hinfo_templates WHERE type=1 ORDER BY pri,hinfo;",
-   addempty=>$hinfo_addempty_mode, empty=>0, iff=>['type','1']},
+  #HINFO appended here if enabled (pos 22)
   {ftype=>1, tag=>'ether', name=>'Ethernet address', type=>'mac', len=>17,
    conv=>'U', iff=>['type','[19]'], empty=>$main::SAURON_RHF{ether}},

@@ -345,6 +374,13 @@
  chr_group=>\$chr_group
 );

+# add HINFO fields in forms (if enabled in config file)
+unless( defined($main::SAURON_DISABLE_HINFO) ? $main::SAURON_DISABLE_HINFO : 0) {
+  splice(@{$host_form{data}}, 20, 0, @hinfo_hf);
+  splice(@{$restricted_host_form{data}}, 12, 0, @hinfo_rhf);
+  splice(@{$new_host_form{data}}, 27, 0, @hinfo_nhf);
+  splice(@{$restricted_new_host_form{data}}, 22, 0, @hinfo_nrhf);
+}

 my %new_alias_form = (
  data=>[
```

Patch 11: import (include support)

Listing C.11: import patch

```
--- a/import    2011-08-07 00:32:08.137363321 +0200
+++ b/import    2011-08-23 17:11:36.000000000 +0200
@@ -51,7 +51,7 @@
 db_connect();
 set_muser($user);

-fatal("cannot open named.conf ($namedf)") unless (-r $namedf);
+fatal("cannot read named.conf ($namedf)") unless (-r $namedf);

 if ($namedf =~ /(^.*\/)/) {
     $dir=$1;
@@ -70,10 +70,23 @@
 ##############################################
 # parse named.conf

-open(NAMEDCONF,"$namedf") || fatal("cannot open named.conf ($namedf)");
-$buf='';
+$namedf =~ /(\/.*\/)/;
+$namedf_base = $1;
+#print "named.conf basedir: $namedf_base\n";
```

```
-while (<NAMEDCONF>) {
+read_file($namedf);
+
+# Refactored into subroutine to allow recursive
+# calls in order to process includes.
+sub read_file {
+
+  $namedtmp = shift;
+
+  open(NAMEDCONF,"$namedtmp") || fatal("cannot open file ($namedtmp)");
+  print "* Reading file: $namedtmp\n";
+  $buf='';
+
+  while (<NAMEDCONF>) {
   chomp;
   s/(\/\/|\#).*$//; # eat one-line comments
   s/\/\*.*\*\///g;  # eat one-line comments
@@ -96,6 +109,16 @@
   s/\s+/\ /g; # eat extra whitespaces
   s/(^\s+|\s+$)//g;

+    # added include support - gbosch 2010-08-22
+    # to avoid problems with bind-chroot, included files are only
+    # treated if they are in the same directory than named.conf
+    if (/include\s+\"(.+)\"/) {
+      $filename = $1;
+      $filename =~ s/^\/.*\///;  #drop filename path
+
+      read_file("$namedf_base$filename");  #namedconf_basedir + filename
+    }
+
   s/{\s*/{\n/g;
   s/;\s*/;\n/g;
   $buf.=$_; $partial='';
@@ -111,9 +134,11 @@
     }
   }
   $buf=$partial;
+  }
+  push @NAMEDCONF, $partial if ($partial);
+  close(NAMEDCONF);
 }
-push @NAMEDCONF, $partial if ($partial);
-close(NAMEDCONF);
+

 push @allow_transfer, [];
```

Patch 12, 13: import-dhcp (includesupport)

Listing C.12: import-dhcp patch

```
--- a/import-dhcp       2011-08-07 00:32:08.137363321 +0200
+++ b/import-dhcp       2011-08-24 12:56:40.000000000 +0200
@@ -56,9 +56,12 @@
      $dir="./";
  }

-# parse named.conf
+$dhcpdf =~ s/^.*\///;    #drop path
+
+# parse dhcpd.conf
  undef %data;
-process_dhcpdconf($dhcpdf,\%data);
+process_dhcpdconf($dhcpdf, $dir, \%data);  #pass the directory of imported dhcpd.conf
+                                           #to look for includes in that directory.

  unless ($opt_notransaction) {
    db_begin();
```

Listing C.13: Sauron/UtilDhcp.pm patch

```
--- a/Sauron/UtilDhcp.pm        2011-08-07 00:32:08.133157526 +0200
+++ b/Sauron/UtilDhcp.pm        2011-08-25 13:09:05.000000000 +0200
@@ -9,6 +9,7 @@
  use Sauron::Util;
  use strict;
  use vars qw($VERSION @ISA @EXPORT);
+use feature "state";

  $VERSION = '$Id: UtilDhcp.pm,v 1.5 2003/03/05 08:14:22 tjko Exp $ ';

@@ -20,16 +21,24 @@

  # parse dhcpd.conf file, build hash of all entries in the file
  #
-sub process_dhcpdconf($$) {
-  my ($filename,$data)=@_;
+sub process_dhcpdconf($$$) {
+  my ($filename, $dir, $data)=@_;

    my $fh = IO::File->new();
-  my ($i,$c,$tmp,$quote,$lend,$fline,$prev,%state);
-
-  print "process_dhcpdconf($filename,DATA)\n" if ($debug);
-
-  fatal("cannot read conf file: $filename") unless (-r $filename);
-  open($fh,$filename) || fatal("cannot open conf file: $filename");
+  my ($i,$c,$tmp,$quote,$lend,$fline,$prev);
+  state %state;    #as used in a recursion call, declare it
+                   #to don't be reinitialized (persistent)
+
+  my $filepath = "$dir$filename";
+  $filepath =~ /(^.*\/)/;
```

```
+  my $dhcpdf_base = $1;
+  print "dhcpd.conf basedir: $dhcpdf_base\n";
+
+  print "process_dhcpdconf($filepath,DATA)\n" if ($debug);
+
+  fatal("cannot read conf file: $filepath") unless (-r $filepath);
+  open($fh,$filepath) || fatal("cannot open conf file: $filepath");
+  print "* Reading file: $filepath\n";

   $tmp='';
   while (<$fh>) {
@@ -41,6 +50,14 @@
 #    print "line '$_'\n";
     s/\s+/\ /g; s/\s+$//; # s/^\s+//;

+    # Add include support - gbosch 2011-08-24
+    # look for included files in the same directory than dhcpd.conf
+    if (/include\s+\"(\S+)\"/) {
+        my $tmpfile = $1;
+        $tmpfile =~ s/^.*\///;    #drop path
+        process_dhcpdconf($tmpfile, $dir, $data);
+    }
+
     for $i (0..length($_)-1) {
       $prev=($i > 0 ? substr($_,$i-1,1) : ' ');
       $c=substr($_,$i,1);
@@ -56,7 +73,7 @@
       }
     }

-    fatal("$filename($.): unterminated quoted string!\n") if ($quote);
+    fatal("$filepath($.): unterminated quoted string!\n") if ($quote);
   }
   process_line($tmp,$data,\%state);
```

Patch 14: LDAP zone storage backend

Listing C.14: sauron patch

```
--- a/sauron    2011-08-07 00:32:08.137363321 +0200
+++ b/sauron    2011-09-01 19:12:53.000000000 +0200
@@ -386,7 +386,14 @@
     $processed_zones{$zonename}++;

 if ($bind_conf) {
+
+    $backend = 'file';  #default storage backend
+
     if ($type eq 'master') {
+      if ($opt_ldap) {
+        $backend = 'database';
+        $zstorage = "ldap $opt_ldap $server{'ttl'}";   # ldap URI TTL
+      }
       $zfile=$server{'pzone_path'};
       $zfile=expand_zonefile_path($zfile,$zonename);
     }
@@ -395,7 +402,9 @@
       $zfile=expand_zonefile_path($zfile,$zonename);
     }
     else { $zfile=''; }
+
     $zfile.=$zonename . ".zone";
+    unless ($zstorage) { $zstorage = $zfile; }

     print BINDFILE "zone \"$zonename\" $zone{class} {\n";
     print BINDFILE "\ttype $type;\n";
@@ -403,7 +412,7 @@
     print BINDFILE "\tcheck-names $val;\n" if ($val ne '');
     $val=$yes_no_enum{$zone{nnotify}};
     print BINDFILE "\tnotify $val;\n" if ($val ne '');
-    print BINDFILE "\tfile \"$zfile\";\n"
+    print BINDFILE "\t$backend \"$zstorage\";\n"
       if ($type eq 'master' || $type eq 'slave');
     print BINDFILE "\tforward $forward_enum{$zone{forward}};\n"
       if ($forward_enum{$zone{forward}} && $type eq 'forward');
@@ -2096,7 +2105,7 @@

 $result=GetOptions("help|h","all|a","bind|b","dhcp|d","printer|p","mail",
                    "updateserial","noupdateserial","verbose","dhcp2","clean",
-                   "check","dhcpclass=s","dhcpvlans=s","tinydns|t");
+                   "check","dhcpclass=s","dhcpvlans=s","tinydns|t","ldap=s");

 if ($opt_help || @ARGV < 1 || $result < 1) {
     print "syntax: $0 [--help] [options] <servername> [<target directory>]\n";
@@ -2104,6 +2113,8 @@
       "\t--all                generate all configuration files\n",
       "\t--bind               generate BIND (named) configuration files\n",
       "\t--tinydns            generate DJBDNS (tinydns) configuration files\n",
+      "\t--ldap=<URI>         generate named.conf/named.zones to use LDAP backend\n",
+      "\t                     e.g. URI: \"ldap://ou=dns,dc=example,dc=com\"\n",
       "\t--dhcp               generate DHCP (dhcpd) configuration files\n",
       "\t--clean              cleanup expired records and vacuum database\n",
       "\t--printer            generate PRINTER (lpd) configuration files\n\n",
```

Appendix D

LDAP Data Interchange Format files (LDIFs)

This appendix is intended to list the LDIF files used in B.3 to load configurations to LDAP server using the RTC (or cn=config) backend.

DNS Base

Listing D.1: dnsbase.ldif

```
# example.com
dn: dc=example,dc=com
objectClass: top
objectClass: dcObject
objectClass: organization
o: Universitat de Lleida
dc: udl

# dns, example.com
dn: ou=dns,dc=example,dc=com
objectClass: organizationalUnit
ou: dns

# internal, dns, example.com
dn: ou=internal,ou=dns,dc=example,dc=com
objectClass: organizationalUnit
ou: internal

# external, dns, example.com
dn: ou=external,ou=dns,dc=example,dc=com
objectClass: organizationalUnit
ou: external
```

DNS indices

Listing D.2: dnsindices.ldif

```
# DNS indices
#
dn: olcDatabase={1}bdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: zoneName eq
add: olcDbIndex
olcDbIndex: relativeDomainName eq
```

TLS server configuration

Listing D.3: tlsconfig.ldif

```
dn: cn=config
changetype: modify
add: olcTLSCertificateFile
olcTLSCertificateFile: /etc/pki/tls/certs/slapd-cert.pem
-
add: olcTLSCertificateKeyFile
olcTLSCertificateKeyFile: /etc/pki/tls/certs/slapd-key.pem
```

Replication

Listing D.4: replicator-user.ldif

```
dn: cn=replicator,dc=udl,dc=cat
objectClass: simpleSecurityObject
objectClass: organizationalRole
cn: replicator
description: LDAP replicator
userPassword: {SSHA}d7DV4HM9gxCjl80pDqttjn/JfWE2eEmr
```

Listing D.5: provider.ldif

```
# Load syncrepl module
#
dn: cn=module,cn=config
objectClass: olcModuleList
cn: module
olcModuleLoad: syncprov
```

```
dn: olcOverlay=syncprov,olcDatabase={1}bdb,cn=config
changetype: add
objectClass: olcOverlayConfig
objectClass: olcSyncProvConfig
olcOverlay: syncprov

dn: olcDatabase={1}bdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryCSN,entryUUID eq
-
add: olcLimits
olcLimits: dn.exact="cn=replicator,dc=example,dc=com"
 size=unlimited time=unlimited
```

Listing D.6: consumer.ldif

```
# syncrepl indices, setup syncrepl
#
dn: olcDatabase={1}bdb,cn=config
changetype: modify
add: olcDbIndex
olcDbIndex: entryUUID eq
-
add: olcSyncRepl
olcSyncRepl: rid=<NUM> provider=ldaps://sauron.example.com bindmethod=simple
 binddn="cn=replicator,dc=example,dc=com" credentials=<PASSWORD>
 searchbase="ou=internal,ou=dns,dc=example,dc=com"
 type=refreshAndPersist retry="60 +"
```

dNSZone schema

Listing D.7: dnszone.ldif

```
dn: cn=dnszone
objectClass: olcSchemaConfig
cn: dnszone
olcAttributeTypes: {0}( 1.3.6.1.4.1.2428.20.0.0 NAME 'dNSTTL' DESC 'An integer
  denoting time to live' EQUALITY integerMatch SYNTAX 1.3.6.1.4.1.1466.115.121
 .1.27 )
olcAttributeTypes: {1}( 1.3.6.1.4.1.2428.20.0.1 NAME 'dNSClass' DESC 'The clas
 s of a resource record' EQUALITY caseIgnoreIA5Match SYNTAX 1.3.6.1.4.1.1466.1
 15.121.1.26 )
olcAttributeTypes: {2}( 1.3.6.1.4.1.2428.20.0.2 NAME 'zoneName' DESC 'The name
  of a zone, i.e. the name of the highest node in the zone' EQUALITY caseIgnor
 eIA5Match SUBSTR caseIgnoreIA5SubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121
 .1.26 )
olcAttributeTypes: {3}( 1.3.6.1.4.1.2428.20.0.3 NAME 'relativeDomainName' DESC
  'The starting labels of a domain name' EQUALITY caseIgnoreIA5Match SUBSTR ca
 seIgnoreIA5SubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {4}( 1.3.6.1.4.1.2428.20.1.12 NAME 'pTRRecord' DESC 'domain
```

```
  name pointer, RFC 1035' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5Subs
 tringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {5}( 1.3.6.1.4.1.2428.20.1.13 NAME 'hInfoRecord' DESC 'host
  information, RFC 1035' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5Subst
 ringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {6}( 1.3.6.1.4.1.2428.20.1.14 NAME 'mInfoRecord' DESC 'mail
 box or mail list information, RFC 1035' EQUALITY caseIgnoreIA5Match SUBSTR ca
 seIgnoreIA5SubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {7}( 1.3.6.1.4.1.2428.20.1.16 NAME 'tXTRecord' DESC 'text s
 tring, RFC 1035' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5SubstringsMa
 tch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {8}( 1.3.6.1.4.1.2428.20.1.18 NAME 'aFSDBRecord' DESC 'for
 AFS Data Base location, RFC 1183' EQUALITY caseIgnoreIA5Match SUBSTR caseIgno
 reIA5SubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {9}( 1.3.6.1.4.1.2428.20.1.24 NAME 'SigRecord' DESC 'Signat
 ure, RFC 2535' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5SubstringsMatc
 h SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {10}( 1.3.6.1.4.1.2428.20.1.25 NAME 'KeyRecord' DESC 'Key,
 RFC 2535' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5SubstringsMatch SYN
 TAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {11}( 1.3.6.1.4.1.2428.20.1.28 NAME 'aAAARecord' DESC 'IPv6
  address, RFC 1886' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5Substring
 sMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {12}( 1.3.6.1.4.1.2428.20.1.29 NAME 'LocRecord' DESC 'Locat
 ion, RFC 1876' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5SubstringsMatc
 h SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {13}( 1.3.6.1.4.1.2428.20.1.30 NAME 'nXTRecord' DESC 'non-e
 xistant, RFC 2535' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5Substrings
 Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {14}( 1.3.6.1.4.1.2428.20.1.33 NAME 'sRVRecord' DESC 'servi
 ce location, RFC 2782' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5Substr
 ingsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {15}( 1.3.6.1.4.1.2428.20.1.35 NAME 'nAPTRRecord' DESC 'Nam
 ing Authority Pointer, RFC 2915' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnor
 eIA5SubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {16}( 1.3.6.1.4.1.2428.20.1.36 NAME 'kXRecord' DESC 'Key Ex
 change Delegation, RFC 2230' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5
 SubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {17}( 1.3.6.1.4.1.2428.20.1.37 NAME 'certRecord' DESC 'cert
 ificate, RFC 2538' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5Substrings
 Match SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {18}( 1.3.6.1.4.1.2428.20.1.38 NAME 'a6Record' DESC 'A6 Rec
 ord Type, RFC 2874' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5Substring
 sMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {19}( 1.3.6.1.4.1.2428.20.1.39 NAME 'dNameRecord' DESC 'Non
 -Terminal DNS Name Redirection, RFC 2672' EQUALITY caseIgnoreIA5Match SUBSTR
 caseIgnoreIA5SubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {20}( 1.3.6.1.4.1.2428.20.1.43 NAME 'dSRecord' DESC 'Delega
 tion Signer, RFC 3658' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5Substr
 ingsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {21}( 1.3.6.1.4.1.2428.20.1.46 NAME 'rRSIGRecord' DESC 'RRS
 IG, RFC 3755' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5SubstringsMatch
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcAttributeTypes: {22}( 1.3.6.1.4.1.2428.20.1.47 NAME 'nSECRecord' DESC 'NSEC
 , RFC 3755' EQUALITY caseIgnoreIA5Match SUBSTR caseIgnoreIA5SubstringsMatch S
 YNTAX 1.3.6.1.4.1.1466.115.121.1.26 )
olcObjectClasses: {0}( 1.3.6.1.4.1.2428.20.3 NAME 'dNSZone' SUP top STRUCTURAL
  MUST ( zoneName $ relativeDomainName ) MAY ( DNSTTL $ DNSClass $ ARecord $ M
 DRecord $ MXRecord $ NSRecord $ SOARecord $ CNAMERecord $ PTRRecord $ HINFORe
 cord $ MINFORecord $ TXTRecord $ SIGRecord $ KEYRecord $ AAAARecord $ LOCReco
 rd $ NXTRecord $ SRVRecord $ NAPTRRecord $ KXRecord $ CERTRecord $ A6Record $
  DNAMERecord ) )
```

Bibliografia

[Ait05] Ron Aitchison. *PRO DNS and BIND*. Apress, 2005.

[AL98] Paul Albitz and Cricket Liu. *DNS and BIND*. O'Reilly and Associates, Inc., 1998.

[Bar96] D. Barr. Common dns operational and configuration errors. RFC 1912, February 1996.

[Ber] Daniel J. Bernstein. Bind, the buggy internet name daemon. http://cr.yp.to/djbdns/blurb/unbind.html (september 2011).

[BLM08] V. Bianco, L. Lavazza, and S. Morasca. Analysis of relevant open source projects and artefacts. quality platform for open source software, 2008.

[Dro93] R. Droms. Dynamic host configuration protocol. RFC 1541, October 1993.

[EBBP97] R. Elz, R. Bush, S. Bradner, and M. Patton. Selection and operation of secondary dns servers. RFC 2182, July 1997.

[EP99] D. Eastlake and A. Panitz. Reserved top level dns names. RFC 2606, June 1999.

[Law02] D. Lawrence. Obsoleting iquery. RFC 3425, November 2002.

[Moc87a] Paul V. Mockapetris. Domain names – concepts and facilities. RFC 1034, November 1987.

[Moc87b] Paul V. Mockapetris. Domain names – implementation and specification. RFC 1035, November 1987.

[Oht96] M. Ohta. Incremental zone transfer in dns. RFC 1995, August 1996.

[ope] Openldap software 2.4 administrator's guide. http://www.openldap.org/doc/admin24/.

[Pos94] J. Postel. Domain Name System structure and delegation. RFC 1591, March 1994.

[RGC03] Gregorio Robles, Jesús M. Gonzalez, and José Centeno. Studying the evolution of libre software projects using publicly available data, 2003.

[Rib09] Josep M. Ribó. Analysis of FOSS projects, 2009.

[sau] Sauron web site. http://sauron.jyu.fi/.

[Ven] Stig Venaas. Ldap sdb back-end for bind 9. http://bind9-ldap.bayour.com/ (september 2011).

[WS06] M. Wong and W. Schlitt. Sender policy framework (spf) for authorizing use of domains in e-mail, version 1. RFC 4408, April 2006.