LABURPENA

Software libreak azken urteetan izan duen garapena nabaria da. Interneteko web edo posta zerbitzarietan aspalditik erabili bada ere, gaur egun erakunde publiko nahiz pribatu, unibertsitate edo enpresatan gero eta gehiago zabaltzen ari da.

Hau honela, eta teknologia berrien gaineko menpekotasunak gora egin duenez, software librea erronka berrien aurrean aurkitzen da, non gero eta azpiegitura fidagarriago, eraginkorrago, malguago, seguruago edo merkeagoak bilatzen diren.

Proiektu honek gaur egungo beharrak software librearen bitartez jorratzeko modua aurkezten du, bideratze, baliabideen esleipen, biltegiratze edo segurtasuna modurik praktikoenean landuz.

HITZ GAKOAK

Sofware librea, konputagailu sareak, Linux, Internet.

AURKIBIDEA

TAULEN AURKIBIDEA

IRUDIEN AURKIBIDEA

ERABILITAKO IRUDIAK

Irudi eta sare diagrama guztiak *Dia* (http://www.gnome.org/projects/dia/) aplikazioarekin egin dira:

Kode eta sasi-kodea `courier` letra motarekin idatzi da. Oharrak "#" ikurrarekin hasten dira. Kodea argiagoa izateko batzuetan "*erabiltzailea@makina:path*#" *prompt*-a kendu da.

```
root@zerbitzari1:/tmp# Kode zatia gehi prompt
Kode zatia prompt gabe
```

1.- SARRERA

Demagun enpresa berri baten aurrean gaudela, edo demagun Internet sekula erabili ez duen altzairugintzan dabilen enpresa baten aurrean gaudela, edo bestela, demagun aspalditik teknologia berrien mundu honetan sartuta dagoen multinazionalak kontratatu gaituela bere azpiegitura informatikoan hobekuntzak egiteko.

Ingurune guzti hauek zeharo desberdinak badira ere, informatikari dagokionean helburu berdintsua konpartitzen dute, alegia, ondo funtzionatzen duen azpiegitura nahi dute. Hots, lana ondo egiten duena, etengabe martxan dagoena, segurua, azkarra, maneiatzen erraza, eta azkenik, baina ez garrantzirik ez duelako, merkea!

1.1.- Lana ondo egiten duen azpiegitura

Ingeniaritzaren hainbat arlotan "lana ondo egitea" hitzaren definizioan bertan egon beharko litzatekeen kontzeptua da. Halere, ez da aparteko lanik egin behar hau beti betetzen ez dela ikusteko. Zein ez da sekula kargatzen ez den web gune baten aurrean egon? Zeinek ez du bakarrik, itxuraz ezer gertatu gabe, gelditu den aplikazio bat ikusi? Zein ez da memoria agortu eta erabat blokeatu den sistema eragile batekin aritu?

Proiektu honetan ez da "aplikazio perfektua" edo arazorik ez duen sistema eragilea bilatzen, baina etengabe ematen diren ezustekoen aurrean zenbait neurri har ditzakegu hauen kaltea txikiagotzeko. Hau da, hain zuzen ere, landuko duguna.

1.2.- Etengabe funtzionatzen duen azpiegitura

Zer egin daiteke zerbitzari bat argindarrik gabe gelditzen bada? Zer egin kable bat askatzen denean? Eta disko gogor bat erretzen bada?

Gaur egungo enpresatan informatikaren beharrak izugarri egin du gora. Gero eta gehiago

dira informatikan erabateko menpekotasuna dutenak bai barne funtzionamendurako bai bezeroekiko harremanetarako.

Azpiegitura fidagarriak egin behar dira, baina hau ez da beti gauza xumea. Beti egongo da balantzaren alde batean fidagarritasuna, eta bestean sinpletasuna edo aurrekontu arazoak.

Nola neurtzen da fidagarritasuna? Ba al da modurik azpiegitura bat ehuneko ehunean sendoa dela esateko? Oinarrizko kalkulu batzuk egin daitezkeela ikusiko dugu arazo hau formalizatzeko, neurtu ahal izateko.

1.3.- Azkarra den azpiegitura

Edozein azpiegitura diseinatzen denean, beharren arabera hardware nahiz software egokia erosten da. Zenbat trafiko izango den, zenbateko prozesu ahalmena behar den edo biltegiratzerako beharko den tokia bezalako galderak arruntak dira erabakiak hartzeko orduan.

Hau honela, sistemaren azkartasuna dugun hardware eta aplikazioekin soilik erlazionatzea izan daiteke ohiko joera. Zalantzarik gabe, hardware azkarragoak eraginkortasunean hobekuntza dakar, baina, ba al da modurik ditugun baliabideak momentuko beharretara optimizatzeko? Eta behin mugara iritsita, ba al da biderik gure azpiegiturari ahalmen handiagoa modu errazean gehitzeko?

1.4.- Segurua den azpiegitura

Informatikaren erabilerak gora egiten duen bezalaxe, segurtasunaren beharrak ere gora egiten du. “Kanpotik” onartu gabeko atzipenen arazoa txikia balitz, gero eta gehiago ematen dira barrutik sortutako arazoak. Besteak beste, posta elektronikoz jasotako birus edo troiako zaldiek izugarrizko kaltea eragin dezake behin sare barruan daudenean. Eta

kontutan izan gaur egungo informatikarekiko menpekotasuna ez dela soilik funtzionala. Sarea birusaren eraginez ordu gutxi batzuetan gelditua egotea eragozpen txikia izan daiteke troiako zaldi baten eraginez bezero guztien datu pribatuak eskuragarri uztearekin alderatuta.

Guzti honengatik, azpiegitura behar bezala kontrolatuta egotea derrigorrezkoa da, bai kanpotik datozen erasoen aurrean, bai barrutik sor daitezkeenen aurrean.

1.5.- Merkea den azpiegitura

Eta nola bideratu orain arteaipatutakoak aurrekontu guztia lanaren erdian agortu gabe?

Software librea[1] erantzun egokia izan daiteke. Egokia software librearen munduan kalitate handiko proiektuak daudelako, eta egokia software librea gehienetan software itxia baino merkeagoa delako. Kontuz! Merkeagoa esaten dudanean ez dut dohain esaten.

Software librean arituko denak prestakuntzarako epe bat behar du, ez bakarrik inplantaziorako, baita mantentzerako ere. Eta denbora dirua dela edozein enpresaburuk ziurtatu dezakeen gauza da.

Software itxiari dagokionean, zenbaitetan hasierako heziketa saltzaileak egiten dituen ikastaroen bitartez egiten da. Ez hori bakarrik, zenbaitetan software enpresaren teknikari baten esku geldi tzen da aplikazioaren martxan jartzea.

Laburbildu ditzadan ezaugarriok taula batean:

1 Software librearen definizioaren inguruan http://www.fsf.org/ eta http://www.opensource.org/docs/osd helbideak interesgarriak izan daitezke.

	Software librea	Software itxia
Kalitatea	*Apache*[2] edo *Mysql*[3] bezalako proiektuak kalitate handikoak dira. Erabiltzaile kopuru itzela dute[4]. Ona dagoen bezala txarrik ere badago.	Software librearekin bezalaxe, ona eta txarra aurki daiteke.
Inplantazioa	Erabiltzailearen esku.	Gehienetan erabiltzailearen esku, baina zenbaitetan saltzailearen kontura.
Mantentzea	Erabiltzailearen esku.	Erabiltzailearen esku edo mantentze kontratuen bitartez.
Menpekotasuna	Gutxi, kodea irekia da, nahi izanez gero aldaketak egiteko.	Handia. Aldatzeko zailtasunak eta inbertsioa amortizatzeko beharra.

Taula 1: Software libre eta itxiaren arteko desberdintasunak

Zein da hobea? Hasieran pentsa daiteke instalazio nahiz mantentzea softwarea egin duenaren esku uztea egokia izan daitekeela, baina hau ez da beti zilegi izaten. Batetik askotan dirutza behar delako kontratu hauek izateko, eta askotan ezin izaten da ziurtatu urtetan mantendu ahal izango denik kontratu hau. Bestetik, praktikan ez da bermatua egoten software itxiarekin arazoak edukiz gero konponbide egokia jasoko denik. Bertsio berria instalatzeko edo martxan dagoen sistema bat zeharo aldatzeko gomendioak maiz entzuten diren soluzioak dira, eta hauek ez dira beti bideragarriak izaten. Azkenik, eta aurrekoarekin erlazionatuta, menpekotasun handia sor daiteke gure eta software itxia

2 http://www.apache.org.

3 Hurrengo atalean, "4. - Ikasketa prozesua" deskribapena eta estekak ematen ditut.

4 *Apache*-ren erabilera datuak esate baterako http://news.netcraft.com helbidean.

egiten duenaren artean. Ez bakarrik softwarearekin, hardwarearekin ere antzeko egoera bat eman daiteke.

Software librearekin, bestalde, nahiz eta ikasketa edo inplantazioa motelagoa izan daitekeen, denborarekin egindakoaren ezagutza handia izaten da. Hau nire ustez garrantzitsua da. Gainera, eta batez ere kalitateko proiektuetaz ari bagara, dokumentazioa barra-barra eskuratu daiteke, Interneten edo bibliografian.

Azken urteotan software irekiaren munduan aintzindari izan diren zenbait proiektuk formula berriak ustiatzen hasi dira. Bi adibide jartzearren, *Sendmail*[5] posta zerbitzari ezaguna ezer ordaindu gabe erabili badaiteke ere, bertsio komertzial bat saltzen du zenbait enpresak dituzten beharretara egokitzeko asmoz. Bestetik, *Dotproject*[6] bezalako proiektuen gestiorako sistemaren garapen taldea kontratatu daiteke softwarea egokitzeko edo modulu bereziak egiteko.

Halere gauzak ez dira beti txuri edo beltz. Egoera bakoitza desberdina da, norberaren esku egongo da momentu bakoitzean erabaki zuzena hartzea, beti ere kontutan izanda teknologia helburu batera iristeko tresna baino ez dela, eta ez helburua bere baitan.

5 http://www.sendmail.org/ eta http://www.sendmail.com/.

6 http://www.dotproject.net/.

2.- PROIEKTUAREN HELBURUEN DOKUMENTUA

Enpresaren munduan, batez ere Interneten dabiltzatenen artean, software librearen erabilera ez da gauza berria. *Netcraft* gunearen datuak[7] kontutan izaten baditugu, *Apache* 1996.etik aurrera izan da web zerbitzarien artean erabiliena. Halere, orain gutxirarte erabilera hau zenbait ingurune konkretutara mugatzen zen. Denborarekin, erabilera hau zabaltzen ari da.

2.1.- Proiektuaren helburuak

Proiektu honen helburu nagusia software libreaz baliatuta gaur egungo behar profesionaletara egokitzen den sare azpiegitura egin daitekeela frogatzeada.

Honetarako, enpresaren beharretara egokitzen den sistema egituratuko dugu; hasieran txikia izan daitekeena, baina beharren arabera hedatuko dena, baliabideak behar bezala esleituz, zerbitzua martxan mantenduz ahal den guztietan, fidagarritasuna eta segurtasuna ahal den neurrian bermatuz eta momentuoro monitorizatuta egongo dena.

Helburuen zerrenda luzea da.

1. Hedagarritasuna lortzea.
2. Baliabideen esleipen egokia izatea.
3. Fidagarritasuna lortzea.
4. Fidagarritasuna neurtu ahal izatea.
5. Segurtasuna lortzea.
6. Sistema kontrolatuta izatea.

Eta guzti hauek sare azpiegitura batean aurkitzen diren atal guztietarako.

1. Bideratze mailan.
2. Segurtasunaren mailan.

7 http://news.netcraft.com/archives/web_server_survey.html

3. Zerbitzari mailan, WWW, SMTP, DNS, datu-base, ...
4. Fitxategi sistema mailan.

Guzti hau lortzea ezinbestekoa da ingurune erreal batean. Proiektu honetan ez ditut atal guztiak sakontasunez landuko. Zenbaitetan erreferentziak emango ditut, nahi duenak hasiera puntu bat izan dezan.

Segurtasunarekin sartzen naizean, esate baterako, ohiko suhesien eredua aipatu baino ez dut egingo. Ez suhesiak garrantzitsuak ez direlako, baizik eta honezkero asko landutako gaia delako. Gaur egun OSI ereduaren maila askotan aurkitu daitezke suhesiak[8], eta bibliografia zabala da. Horregatik, emango ditudan ideiak sareen monitorizazioari begira izango dira (Network Security Monitoring[9]), gutxiago landu den gaia baita.

2.2.- Lan Metodologia

Proiektuaren definizio eta helburuak finkatu ostean, aurrera eramateko modurik onena proiektuari ahalik eta ikuspunturik praktikoena ematea dela uste dut. Hau lortzeko biderik egokiena kontsultoria moduko metodologia jarraitzea dela iruditzen zait. Baina, bestalde, ez dut enpresa konkretu baten egoerara mugatu nahi. Orokortasuna nahi dut, kasuistika zabala landu nahi dut, nahiz eta planteatuko ditudan egoera guztiak ez diren izango enpresa guztietan aplikagarri.

Hau honela, "Fnnet" izeneko ustezko enpresa erabikiko dut. Hasieran, gure ingurunean egon daitekeen edozein enpresa txikiren oinarrizko azpiegituraren antzekoa izango du, eta gero hainbat hobekuntza planteatuko ditut oinarri honetatik abiatuta.

Eskematikoki, garapen prosezua jarraiko zerrendarenitxurakoa izango da:

8 (Bridge, 2006) 2.mailan, (Netfilter, 2006) 3. eta 4.etan (gehiagotan ere erabilgarria da), eta 7.mailarako, web ingurunetan, *Apache*-ren moduluren bat, esate baterako. Modulu honen adibidea http://www.modsecurity.org helbidean.

9 (Tao, 2005) Liburua gomendagarria da, Interneten aurkitu daitekeen informazioarekin batera.

1. Oinarrizko azpiegituran deskribapena. Aurreikusitako arazoak eta konponbideak.
2. Hobekuntzak bideratzean.
3. Hobekuntzak segurtasunean.
4. Hobekuntzak fidagarritasunean.
5. Hobekuntzak baliabideen esleipenean.
6. Hobekuntzak azpiegituraren monitorizazioan.
7. Hobekuntzak datuen biltegiratze mailan.
8. Hobekuntzak datu-base sistematan.

2.3.- Arriskuen analisia

OHARRA – Interesik gabe. Norberak bereak idatzi beharko lituzke hemen.

2.4.- Aurreikusitako baliabideak

Proiektua burutzeko baliabideak hardwareari dagokionean:

1. Makina bat bezeroaren lana egiteko, hots, probak egiteko.
2. Bi makina atal desberdinak lantzeko. Batez ere fidagarritasunari dagozkionak.
3. Makina bat zerbitzarien lana egiteko, HTTP, FTP, SMTP, ... Zorionez, birtualizazioak[10] makina fisiko bakar batean hainbat zerbitzari jartzeko aukera ematen du.
4. Sare hardwareari dagokionean, *switch*[11] txiki batekin nahiko izango da. Edozein kasutan, momenturen batean interesgarri irudituko balitzait zeozer konplexuagoa, sare simuladorea erabiliko dut.

Baliabideak softwareari dagokionean:

10 http://linux-vserver.org/ erabiliko dut, baina ez da aukera bakarra.

11 Hitza ia beti erabiltzen da ingeleraz, zuzenean. http://en.wikipedia.org/wiki/Networking_switch

1. Batez ere Linux sistema eragilea eta *Debian*[12] banaketa.
2. Zerbitzariak. Proiektuan agertu ahala behar bezala erreferentziatuko ditut. Normalean aurrez konpilatutako softwarea erabiliko dut.
3. Aurrez konpilatutako kodea eskuragarri ez dagoenean *Debian* sistemaren gainean konpilatuko dut erabili ahal izateko.

Aipatutako guztia eskuragarri dut aparteko zailtasunik gabe.

2.5.- Lanaren deskonposaketa

Proiektuan egindako lanaren deskonposaketa hainbat fasetan banatuta egongo da. Orokorrean planifikazio, garapen, proba eta dokumentazio lanak izango dira.Azkeneko bi hauek beste faseetan ere tartekatuko ditut, eta ez bakarrik bukaeran.

OHARRA – Norberak aurreikusi behar du zenbat ordu beharko duen.

2.5.1.- Planifikazioa

Fase honetan proiektuaren zehaztapena egingo da. Honekin batera aurrekarien azterketa eta proiektuaren helburuen dokumentua ere idatziko dut.

OHARRA – Ordu kopurua?

2.5.2.- Ikasketa prozesua, softwarearen aukeraketa

Erabiliko dudan software gehiena ezaguna badut ere, badaude gehiegi ezagutzen ez dutudan eta erabili nahiko nituzkeen software librean oinarritutako proiektuak. Hasieran, beraz, egun batzuk beharko ditut hauek ezagutzeko.

12 (Debian, 2006)

Ez hori bakarrik, erabiliko dudan softwarea behar bezala identifikatu eta dokumentatuko dut fase honetan.

OHARRA – Ordu kopurua?

2.5.3.- Ingurunearen prestakuntza

Lanean hasi baino lehen oinarrizko azpiegitura abian jarri beharko dut. Ditudan baliabideak nahikoak badira ere, garapen fasean aurrera egin ahala berrerabili beharko ditut. Aldaketa hauek egiteko ere denbora beharko dut.

OHARRA – Ordu kopurua?

2.5.4.- Garapena

Fase honetan "Fnnet" enpresaren oinarrizko azpiegituraren deskribapena egingo da. Gero, urratsez urrats, hurrengo zerrendan agertzen diren hobekuntzak planteatuko ditut hasierako sistematik abiatuta.

1. Bideratzea hobetu. N+1[13] eta *Standby*[14] sistemak.
2. Bideratzea hobetu. HSRP/VRRP[15], BGP[16] eta gainontzeko protokoloen deskribapen laburra.
3. Suhesi sistema hobetu. VRRP, N+1 eta *Standby* sistemen luzapena.
4. Baliabideen esleipena eta fidagarritasunaren hobekuntza. Karga banatzeko sistemaren deskribapena eta hasierako inplantazioa.
5. Baliabideen esleipena eta fidagarritasunaren hobekuntza. Karga banatzeko sistemaren garapena. VRRP, *Standby*, N+1.

13 "4.2.- Fidagarritasunaren neurketa" atalean azaltzen dut.

14 http://en.wikipedia.org/wiki/Standby_%28telecommunications%29.

15 Proiektuan zehar protokolo hauek zertan oinarritzen diren azalduko dut.

16 http://en.wikipedia.org/wiki/Bgp

6. Bideratze, suhesi eta karga banatzearen ikuspuntu globala.
7. Datu-base sistemak. *Master/Slave* eta *clustering*[17]. Karga banatzea eta datu-base kudeaketa-sistemak.
8. Hobekuntzak fitxategi sistematan. Kopia arrunta nahiz erreplikazioa.
9. Hobekuntzak monitorizazioan. Sistemaren baliabideen monitorizazioa SNMP protokoloaren laguntzaz.
10. Hobekuntzak segurtasunean. Sarearen monitorizazioa segurtasunaren ikuspuntutik.

Fase guztietan ez dut denbora berdina erabiliko. Zenbait fase elkarren artean erlazionatuta daude, edo kontzeptu berdintsuetan oinarritzen dira. Beste batzuk aipatu baino ez ditut egingo, eta oinarrizko bibliografia emango dut nahi duenak gaian sakondu dezan.

OHARRA – Ordu kopurua?

2.5.5.- Probak

Garapenean zehar planteatzen dudan hobekuntza bakoitzeko zenbait proba egingo ditut, emaitzak behar bezala isladatuz proiektuaren memorian. Fase hau, beraz, gainontzekoekin tartekatuta egongo da.

OHARRA – Ordu kopurua?

2.5.6.- Dokumentazioa

Ahal dudan neurrian ez dut dokumentazio guztia bukaerarako utziko. Probekin bezala, fase hau ere gainontzekoekin tartekatuta egongo da. Idatzitako kapitulu guztiak bateratzea, azkeneko ondorioak, denboren desbiderapenaren analisia eta aurkezpenaren prestakuntza izango da bukaerarako lagako dudan bakarra.

17 http://en.wikipedia.org/wiki/Computer_cluster

OHARRA – Ordu kopurua?

3.- AURREKARIEN AZTERKETA

Gaur egungo azpiegitura informatikoek dituzten beharretara egokitzeko software libreak hamaika aukera eskaintzen duela ikusiko dugu proiektu honetan. Esan beharrik ez dago industriak ere katalogo zabala duela behar guzti hauei konponbidea emateko: hardwarean oinarritutako bideragailuak (*Cisco Systems*, *Juniper Networks*, ...), suhesiak eta segurtasunerako tresnak (*Cisco Systems*, *Check Point Software Technologies*, ...), karga banatzeko sistemak (*Radware*, *F5 Networks*, ...), datu-base kudeaketa-sistemak (*Oracle*, IBM, ...), biltegiratzerako hardwarea (EMC, IBM, ...). Azpiegitura bat osatzeko behar den guztia dagoeneko merkatuan aurkitu daiteke.

Software libreak ere proiektu honetan landuko diren atal desberdinak jorratzeko eskaintzen duen katalogoa zabala da. Batetik sistema eragile (*Linux*, *FreeBSD*, ...) eta aplikazio kopuru handia (*Mysql* eta *Postgresql* datu-base kudeaketa-sistema batzuk aipatzearren), eta bestetik arlo konkretuak lantzeko bereziki eraikitako aplikazio-multzoak, hala nola segurtasunari begira sortutako *Engarde Secure Linux* eta *Pfsense*. Aipagarria da, halaber, aplikazioen konfigurazioa errazago egiteko aplikazioak ere badaudela. Azken hauen adibidea *Ultramonkey* deritzona da, proiektuan zehar landuko den karga banatzea eta fidagarritasuna "sinpleago" egiteko erabiltzen delarik.

Linux sistema eragilearen gainean ere hamaika banaketa sortu da, batzuk doan banatzen direnak (*Debian*, *Fedora*, *CentOS*, ...), beste batzuk ordainduta (*SuSE*, *Red Hat Enterprise Linux*, ...). Zerrenda ia amaituezina da.

Erakunde publiko nahiz pribatu, enpresa txiki nahiz handi askok, goian aipatuko software eta hardwarea erabiltzen dute azpiegitura informatikoei fidagarritasun, eraginkortasun, malgutasun edo hedagarritasun handiagoa emateko. Batzuk software librea erabiltzen dute, beste askok itxia, baina aipatutako helburuak berdinak dira.

Proiektuan zehar sareen fidagarritasunaren inguruan hainbat kontzeptu azalduko dira. "*http://www.cisco.com/global/EMEA/ciscoitatwork/pdf/how_cisco_it_achieved_a_highly*

_available_wide_area_network.pdf" helbidean *Cisco Systems* enpresak bere azpiegituraren bederatzien kopurua (fidagarritasuna adierazteko erabili ohi den neurria) handitzeko prozesuaren deskribapena egiten du. Proiektu honetan azpiegitura txikiagoen inguruan arituko banaiz ere, kontzeptu gehienak bai multinazional batean bai Euskal Herriko enpresa txiki batean ere dira aplikagarri.

Laurogeita hamargarren hamarkadan web inguruneak, Interneteko atariak, agertzen hasi ziren. Urte gutxiren buruan hauen erabilpenak gora egin zuen, gero eta garrantzi handiagoa zuten. Egoera berri honekin bai software librean, bai software itxian, bai hardwarean oinarritutako karga banatzeko sistemak agertu ziren. Proiektu honetan aztertuko dudan softwarea garai horretako zenbait atari garrantzitsuetan erabiltzen hasi zen, "*http://www.linuxvirtualserver.org/deployment.html*" helbidean ikusten denez. Baina, esan bezala, software librea ez zen garai horietan lehengo urratsak ematen ziharduen bakarra; IBMek 1996. urtean egindako Kasparov eta *Deep Blue* makinaren arteko xake jokoan karga banatzeko sistema bat erabili zuen momentuko atzipen marka guztiak apurtzeko (*http://www.research.ibm.com/deepblue/learn/html/index.shtml*). Hamarkada bat pasa bada ere, teknologia aldatu bada ere, orduan egindakoaren ideia nagusia gaur egun ere da erabilgarri.

Teknologia garatu ahala eduki askoz dinamikoagoak sortzen hasi ziren. Datu-base kudeaketa-sistema ahaltsuagoak behar ziren, biltegiratze-azpiegitura hobeagoak, ... Honela, hainbat erakunde eta enpresek proiektu honetan deskribatuko dudan bidea hasi zuten, software librearekin, itxiarekin edo bien nahasketarekin. Adibide bat "*http://danga.com/words/2005_oscon/oscon-2005.pdf*" helbidean ikus daiteke.

4.- IKASKETA PROZESUA

Informatikarion lanik gogorrenetakoa software nahiz hardwarearen aukeraketa da. Egoera bakoitza aztertu behar da, eta modu egokian erabaki zein den, beharren arabera, gure ingurunera ondoen egokitzen dena. Atal honen lehenengo zatian, lanarekin aurrera jarraitzeko beharrezko izango ditugun sistema eragileak, bideratze protokoloak inplementatzen dituzten aplikazioak, web zerbitzariak, datu-base kudeaketa-sistemak eta software librean oinarritutako beste zenbait proiekturen zerrenda egingo dut, bakoitzaren deskribapen laburrarekin. Bigarren azpiatalean fidagarritasuna neurtzeko erabili daitekeen metodo baten deskribapena egingo dut.

4.1.- Erabilitako softwarea

4.1.1.- Sistema eragileak

4.1.1.1.- Linux

Linux da zalantzarik gabe software libreari dagokionean gehien erabiltzen den sistema eragilea. Linux aipatzen denean ez dugu nahastu behar ingeleraz *kernel*[18] deritzona eta honen gainean eraikitzen diren banaketak. Proiektu honetan *Debian* erabiliko dut (4.1.1.3 azpiatala).

Gaur egun kernelari dagokionean bi bertsio nagusi ditugu, 2.4 eta 2.6. Proiektu honetan, besterik esan ezean, 2.6 erabiliko dut.

Kernela lortzeko eta informazio gehiagorako (Kernel, 2006).

18 Nukleo da euskaltermen http://www1.euskadi.net/euskalterm/indice_e.htm itzulpena.

4.1.1.2.- FreeBSD

FreeBSD eta bere familiakoak diren *openBSD* eta *netBSD* kalitate handiko sistema eragileak dira. Ez ditut proiektuan erabiliko, ez bada zerbait egiteko modu desberdinak daudela adierazteko, baina oso kontutan izateko sistemak dira hauek.

Informazio gehiagorako (Freebsd, 2006).

4.1.1.3.- Debian

Debian proiektu honetan erabiliko dudan Linux banaketa da. Banaketa bakoitzak bere instalazio sistema izaten du, bere hardwarea detektatzeko tresna, bere fitxategi sistemaren egitura, bere softwarea instalatu eta maneiatzeko tresna, ... baina orokorrean, kernel, liburutegi eta aplikazioak berdintsuak izaten dira beti, nahiz eta batzuetan aldaketa txikiren bat badagoen. Hau honela, hemen *Debian* erabiliko badut ere, aipatuko ditudan gehienak berdin egin daitezke beste edozein banaketan.

Debian nolabaiteko "azpi-banaketetan" sailkatzen da. Nire kasuan *testing* izenekoa erabiliko dut, baina normalean enpresa munduan *stable* deritzona erabiltzea gomendatzen da.

Informazio gehiagorako (Debian, 2006).

4.1.1.4.- Cisco IOS

Cisco Internetwork Operating System hau ez da inolaz ere software librearen barruan sartzen, baina derrigor aipatu behar dudan sistema da, edozein sare azpiegituratan konfiguratu behar den hardwarea dagoelako, eta hardwarearen munduan *Cisco Systems* fabrikatzailearen gailuak erabilietakoenen artean daudelako. Ez da dagoen bakarra, baina hau izan da nire aukera behar izanez gero adibideetan erabiltzeko.

Sareko gailu baten sistema eragileak egiten dituen atazen oinarri oinarrizko zerrenda:

1. Sare protokolo eta funtzioen inplementazioa.
2. Banda zabalera handiko gailuak modu egokian konektatzea.
3. Segurtasuna gehitzea.
4. Hedagarritasuna gehitzea.
5. Fidagarritasuna gehitzea.

Informazio gehiagorako (Cisco, 2006).

4.1.2.- Bideratzea eta sare interfazeak maneiatzeko softwarea

Sare azpiegitura bat diseinatzen denean ezinbestekoa da bideratzea nola egingo den aurretik planifikatuta izatea. Erabaki hauek bai Internetera begira bai gure azpiegituraren barruan dagoen sare diseinuari begira egin beharko dira. Zenbaitetan bideratze estatikoa erabili beharko da, besteetan, ordea, protokoloren bat inplementatzen duen softwarea beharko dugu.

Fidagarritasunari begira ere hartutako erabakiek garrantzia dute. Bideraketa dinamikoki egiten bada, sistemak automatikoki detektatuko ditu sarean ematen diren aldaketak, eta normalean protokoloan definitutako konbergentziarako denbora igarotakoan bideratze taulak berregingo ditu. Bideratzea estatikoa bada, beste tresna batzuk beharko ditugu sistema "adimentsu" bihurtzeko.

4.1.2.1.- Zebra, Quagga

Orain gutxirarte software librearen munduan bideratze protokoloen inplementazioaz galdetuz gero zuzenean *Zebra* izaten zen erantzuna. Halere, proiektu honen garapena gaur egun geldituta dago, eta bere ordez *Quagga* izenekoa sortu da.

Quagga IPv4 eta IPv6 protokoloen gainean lan egiten duen bideratze softwarea da.

Diseinu modularra dauka. Bere osagai nagusiak:

1. *Zebra* deabrua, sistema eragilea eta gainontzeko moduluen artean komunikazioa ahalbideratzeko.
2. *Ospfd*, OSPFv2 protokoloa inplementatzen duena.
3. *Ospf6d*, OSPFv3 protokoloa erabiltzeko.
4. *Ripd*, RIPv1 eta RIPv2 protokoloetarako.
5. *Ripngd*, RIPng-ren inplementazioa.
6. *Bgpd*, BGPv4+ protokoloa inplementatzen duena.

Informazio gehiagorako (Zebra, 2003) eta (Quagga, 2006). Protokoloen RFC dokumentuak (Rfc, 2007) helbidean bila daitezke.

4.1.2.2.- Heartbeat

Softwarea baino kontzeptu batez ari gara heartbeat edo taupada hitza erabiltzen dugunean. Teknologia honen bitartez, hainbat zerbitzarik elkarri "taupada" moduko seinaleak bidaliko dizkiote. Honela, sistema osoaren egoeran ematen diren aldaketak detektatuko dituzte, eta behar bezalako ekintzak burutu. Adibidez, makina bat gelditzen denean, eta taupada seinalerik ez duenean bidaltzen, beste zerbitzari batek egoera hori detektatu eta jausitakoaren IP helbidea bere gain hartuko du.

Informazio gehiagorako (Linux-ha, 2005).

4.1.2.3.- VRRP

VRRP protokoaren Linux gaineko inplementaziorik erabilienetakoa gero aipatuko dudan *Keepalived* softwarearekin batera banatzen da. VRRP ez da bideratze protokolo bat. Bere tokia hainbestetan ikusten den bideratze estatikoaren atebide lehenetsiari[19] fidagarritasun

19 Ingeleraz default gateway. Proiektu honetan zehar gateway hitza ere erabiliko dut, askotan irakutzen den terminoa delako.

handiagoa ematean dago.

Informazio gehiagorako (Keepalived, 2006) eta (Rfc3768, 2004).

4.1.3.- Karga banatzeko sistemak

Karga banatzeko sistemek hiru onura nagusi dakartzate. Batetik, zerbitzua hainbat makinen artean banatzen denez, fidagarritasuna handitzen da. Honekin batera, sistema hedagarriagoa da, dinamikoki, geldiunerik gabe, zerbitzariak gehitu dakizkiokelako azpiegiturari. Azkenik, baliabideen esleipena ere hobetzen da. Demagun karga banatzeko sistema batean Eusko Jaurlaritzaren administrazioaren online zerbitzuak daudela. Demagun bost zerbitzari erabiltzen direla sail bakoitzeko, hala nola, industria saila, osasun saila, etxebizitza eta gizarte saila, ..., eta normalean sistema bere mugaren ehuneko berrogeian dagoela. Suposatu dezagun babes ofizialeko pisuen zozketa bat dagoela, eta etxebizitza sailaren zerbitzariek ohiko atzipen kopurua hirukoiztuko dutela. Karga banatzeko sistemei esker, posible litzateke beste sailetarako aurreikusitako baliabideen parte bat aparteko lanik gabe etxebizitza sailari uztea, eta behin zozketa bukatuta, berriz jatorriko egoerara itzultzea.

4.1.3.1.- Linux Virtual Server

LVS softwarearen garapena 1998an hasi zen. Orain gutxirarte erabili ahal izateko kernelean norberak gehitu beharreko kodea zen; gaur egun, berriz, kernel estandarraren parte da (2.4.23-tik aurrera). Hau honela, erabili ahal izateko behar den bakarra, hasteko Linux nukleo bat da, eta honekin komunikatzeko aplikazio txiki bat.

Honelako sistematan bi elementu berri agertzen dira. Zuzendariak[20] izango du trafikoa jaso, eta esleipen algoritmo baten arabera eskaerak zerbitzari “erreal”[21] batera edo bestera

20 Ingeleraz “director”.

21 Ingeleraz “real server”.

bidaltzearen ardura.

Karga banatzeko sistema bat ezartzen denean ondo jakin behar da zer den egin nahi duguna, eta zein den *load balancing* sistemaren "azpian" egongo den aplikazioa. Merkataritza elektronikoan, esate baterako, bezero batek saio bat irekitzen duenean web zerbitzari "errealarekin" nolabait ziurtatu beharko dugu konexio hau denboran zehar mantentzen dela[22]. Modu desberdinetan egin badaiteke ere, aukera bat bezero horren atzipen guztiak denbora tarte batean zerbitzari berdinera bidaltzea izango litzateke.

Azalpen gehiago garapen fasean emango dut, baina LVS softwareak duen ahalmena erakusteko zuzendariak erabakiak hartzeko dituen esleipen-algoritmo batzuen zerrenda jartzea egokia iruditzen zait:

1. *Round-robin* (RR): Jasotako eskaera bakoitza sekuentzialki banatzen da zerbitzari errealen artean.
2. *Weighted round-robin* (WRR): Zerbitzari bakoitzak pisu bat dauka. Honela, pisua bikoa duen zerbitzariak batekoa duenaren trafikoaren bikoitza jasoko luke.
3. *Destination hashing*: Metodo honen bitartez IP batera doazen eskaerak beti joango dira zerbitzari berdinera.
4. *Source hashing*: Metodo hau erabiltzen da batez ere zuzendariak ziurtatu behar duenean jasotako eskaera bakoitza etorritako bide berdinetik itzuli behar dela. Hau da, zuzendaria bi suhesi edo bideragailuetara konektatuta dagoenean.
5. *Least-Connection* (LC): Eskaera bat jasotakoan, zuzendariak ikusiko du zein zerbitzarik dituen konexio gutxien irekita[23], eta konexio berria horra bidaliko du.
6. *Weighted least-connection* (WLC): Metodo honek LCri pisua gehitzen dio.

Beste metodo asko daude (SED, NQ, LBLC, LBLCR), eta noizean behin berriak gehitzen

22 Karga banatze sisteman ziurtatu daiteke bezero baten konexioa beti zerbitzari berdinera joango dela. Beste ohiko aukera da saioen informazioa datu-baseetan gordetzea, zerbitzari guztiek eskuragarri izateko.

23 ESTABLISHED egoeran

dira.

Informazio gehiagorako (Lvs, 2006), (Kopper, 2005).

4.1.3.2.- Ldirector

Karga banatzeko sistema bat instalatzen denean fidagarritasun, hedagarritasun eta malgutasunaren helburuak nolabait beteta ditugu. Halere, narbarmena egiten da arazoak ere badaudela. Nola detektatuko du zuzendariak zerbitzari erreal bat hondatu dela? Zeintzuk dira egoera honetan hartuko dituen neurriak?

Honi aurre egiteko *Ldirector* eta honezkero aipatu dudan *Keepalived* agertzen dira, besteak-beste. Proiektu honetan, eta bat aukeratzekotan, *Keepalived* erabiliko dut, baina *Ldirector* historikoki oso erabilia izan da *Heartbeat*-ekin batera, eta aipamena merezi duela iruditu zait.

Informazio gehiagorako (Ldirector, 2002).

4.1.3.3.- Keepalived

Aurretik aipatu dudan bezala, *Keepalived* softwareak VRRP protokoloaren inplementazio interesgarria dauka zuzendarien artean fidagarritasuna bermatzeko, baina ez da hau aplikazio honek duen helburu bakarra. Zerbitzari errealak ere uneoro izango ditu kontrolatuta, eta automatikoki aldatu ahal izango du zuzendariaren konfigurazioa edozein gertaeraren aurrean.

Informazio gehiagorako (Keepalived, 2006).

4.1.4.- Fitxategi sistemak. Biltegiratzearen kudeaketa

Karga banatzeko sistemak agertu aurretik, Interneten erabilpena zabaldu aurretik, ohikoa zen HTTP bidez zerbitzatzen zen eduki guztia web zerbitzari bakarrean egotea. Egoera honetan zeuden arazo nagusienak biltegiratzean hardware edo software bidezko fidagarritasuna lortzea[24], fitxategi sistemarik egokienaren aukeraketa eta segurtasu kopiak egiteko modua ziren. Hondamendiaren aurrean, zerbitzari berria instalatuko litzateke, eta segurtasun kopia bidez edukia berreskuratuko genuke.

Gaur egun, ordea, makina asko sartzen dira azpiegituretan, eta biltegiratzearena are eta arazo handiagoa bilakatu da. Web zerbitzarien adibidearekin jarraituta, orain zerbitzari asko izango ditugu, eta guztiek izan beharko lukete edukia ikusteko aukera. Ez hori bakarrik; web gune dinamikoen kasuan, makina batean egindako aldaketak gainontzekoek momentuan ikusi beharko lituzkete.

Hau honela, eta beti ere gure aplikazioaren beharren arabera, askotan biltegiratzea zentralizatu behar dela ondorioztatzen da. NAS[25] bezala ezagututako teknologia izan da hau lortzeko bidea, baina azken urteotan SAN[26] bezala ezagututakoak garrantzia hartu du.

4.1.4.1.- Rsync

Rsync zerbitzari batean dagoen edukia beste batera kopiatzeko tresna da. Aldatutakoa baino ez duela kopiatzen da bere abantailarik nagusiena.

Tresna hau egokia izan daiteke batez ere edukia aldatzen ez denean, web zerbitzari estatikoetan, esate baterako, edo onargarria denean makina batean egindako aldaketak gainontzeko zerbitzarietan berehala ez ikustea. Segurtasun kopietarako ere software

24 RAID sistemekin, adibidez. http://en.wikipedia.org/wiki/RAID

25 Network Attached Storage

26 Storage Area Network

interesgarria da.

Informazio gehiagorako (Rsync, 2006).

4.1.4.2.- NFS

NAS sistema baten oinarrizko ideia sinplea da. Biltegiratzeaz arduratzen den zerbitzari batek edukia sarean atzigarri uzten du, gehienetan NFS eta SMB/CIFS[27] protokoloak erabiliz. Gehiegi sinplifikatzeko arriskuarekin esan genezake funtsean NFS zerbitzaria baino ez dela.

Linux sistema eragileak behar dugun guztia eskaintzen digu NFS zerbitzari bat martxan jartzeko, bai NFSv4 eta NFSv3 bertsioetarako (NFSv2 ez dut aipatuko, oso zaharra baita).

Informazio gehiagorako (Rfc3530, 2003) eta (Rfc1813, 1995).

4.1.4.3.- DRBD

Edukia zentralizatuta izateak onurak baditu ere, arazo argi bat dauka. Zer gertatzen da NFS zerbitzaria hondatzen bada? Segurtasun kopiak derrigorrezkoak dira, noski, baina zenbaitetan, eta batez ere datu asko dagoenean, hondamendiaren aurrean kopia berreskuratzea eta sistema egoera arruntara[28] itzultzea lan gogorra bezain luzea izan daiteke. Are eta gehiago, kontutan izanda enpresa batean zein animo egoten den zerbitzua etenda dagoenean.

DRBD[29] softwareaz makina "nagusian" idatzitakoa beste makina batean automatikoki eta ia denbora errealean kopiatzen da TCP/IP sare arrunta erabiliz. Esan beharrik ez dago

27 Server Message Block (SMB) eta Common Internet File System (CIFS).

28 "Egoera arrunta" proiektuan zehar erabili eta azalduko den kontzeptua da.

29 Distributed Replicated Block Device.

fidagarritasuna asko hobetzen delahonekin.

DRBDek bertsio komertziala ere badu, DRBD+ izenekoa. Azken honek open source bertsioak dituen zenbait muga apurtzen ditu. Proiektu honetan ez ditugu hobekuntza hauek behar, beraz DRBD arrunta, librea, erabiliko dugu.

Open source bertsioaren informazio gehiago jasotzeko (Linux-ha, 2005), (Drbd, 2006) eta (Linbit, 2006).

Bertsio komertzialaren informazioajasotzeko (Linbit, 2006).

4.1.4.4.- Fitxategi sistemak. LVM, Ext3, Reiserfs

Linux kernelean eskuragarri ditugun fitxategi sistemen kopurua handia da. Hauetako bakoitzak aldekoak eta kontrakoak ditu; batzuk fitxategi txikiekin moldatzen dira ondoen, beste batzuk irakurketak optimizatuta dituzte, ... Fitxategi sistemen gaineko eztabaida ez da proiektu honen helburua. Nik *Ext3* eta *Reiserfs* erabiliko ditut, baina, esan bezala, ingurune erreal batean aukeraketa egokia egiteko zer biltegiratuko den jakin behar da.

Azken urteotan, software librearen munduan, fitxategi sistema eta biltegiratzearen inguruan kontzeptu berriak agertu dira. Hauetako batzuk teknologia berrietan oinarritzen dira, beste batzuk orain arte sistema eragile itxietan baino ez zeuden eskuragarri. Honen adibidea LVM[30] deritzona da.

LVM agertu baino lehen fitxategi sistema bat txiki gelditzen zenean ohiko prozesua zen disko gogor berriak erostea, partizioak edo RAID[31] sistemak hutsetik berregitea eta segurtasun kopia bat erabiliz datuak sistema berrira kopiatzea. LVMk honi buelta eman dio; orain eragiketa hauek askoz gardenago eta azkarragoak dira.

30 Logical Volume Manager

31 http://en.wikipedia.org/wiki/RAID

Proiektu honetan ez dut LVM erabiliko, baina enpresa ingurunetan tresna egokia izan daitekeenez aipatzea zilegi iruditzen zait.

Informazio gehiagorako kernelaren dokumentazioa, (Lvm, 2006) eta (Reiser, 2006) hasiera ona izan daiteke.

4.1.5.- Datu-base kudeaketa-sistemak

Datu-base kudeaketa-sistemen erabilera ezinbestekotzat jo dezakegu gaur egungo ia edozein azpiegitura informatikotan. Hau honela, gure helburuetako bat DBKSetan ere fidagarritasuna edo eraginkortasuna bezalakoak hobetzea izan beharko da, noski.

Ikusiko dugunez, datu-base kudeaketa-sistemekin aipatutako hobekuntzak egiteko fabrikatzaile bakoitzak eskainitako metodoak erabiliko ditugu, sarearen beste ataletan erabilitakoak baztertuz. Zorionez, software librean ezagutzen ditugun hainbat sistemekin hau erraz lortuko dugu.

4.1.5.1.- Postgresql

Hamabost urte baino gehiago dira aplikazio hau software librearen munduan agertu zenetik. Gaur egun, kalitate handiko datu-base kudeaketa-sistema da, osoa, ANSI-SQL 92/99 estandarrak betetzen dituena.

Proiektu honetan *Mysql* erabiliko dut, eta ez *Postgresql*, baina baterako aipatuko ditudan ideia gehienak bestearekin erebideragarriak direla adierazi nahi nuke.

Informazio gehiagorako(Postgresql, 2006).

4.1.5.2.- Mysql

Mysql da, seguraski, software librearen ingurunetan gehien erabiltzen den datu-base kudeaketa-sistema. Hasieran sinplea zen, ez zituen tresna komertzial edo *Postgresql* bezalakoen aukera guztiak eskaintzen, baina egiten zuena ondo eta azkar egiten zuen. Honi esker, batez ere web ingurunetan, izugarrizko arrakasta lortu zuen. Azken urteotan funtzionalitatearen aldetik aurrerapauso handiak eman baditu ere, proiektuan zehar fidagarritasunaren inguruan baino ez naiz arituko.

Informazio gehiagorako (Mysql, 2006).

4.1.6.- Segurtasunerako tresnak

Internetera begira dagoen edozein azpiegitura seriotasunez lantzeko segurtasunari garrantzi handia eman beharko genioke. Hau, teorian guztiok ikusten dugu argi, baina gero ez da horrenbestetan praktikara eramaten. Hots, praktikan sare-segurtasuna askotan suhesietara mugatzen da; eta hau, nire ustez, ez da nahikoa. Zergatik? Batetik suhesi perfektua oraindik ez delako asmatu, are eta gutxiago suhesian zehar web, FTP, IRC edo bestelako trafikoa onartu behar dugunean, eta bestetik eraso guztiak ez direlako beti gure saretik kanpo hasten.

Orokorrean, sarea gauza izan behar da erasotaz babesteko (suhesiak), baina gauza izan behar da, bederen, denbora errealean sarean zerbait "arraroa" gertatzen ari dela jakiteko, eta epe laburrean arraroa den hori identifikatzeko. Honela, arazoak zeintzuk diren jakinda, neurriak errezago hartuko ditugu, edo, behar izanez gero, berreskuratze prozesua azkarragoa izango da. Kontzeptu honetan sakontzeko (Tao, 2005) eta (Extrusion. 2006) oso liburu gomendarriak dira.

4.1.6.1.- Netfilter

Netfilter da Linuxeko kernelean dagoen suhesiaren izena (2.4 bertsioaz geroztik). Berau administratzeko *iptables* tresna ezaguna erabiltzen da. Bere ezaugarrien artean aipagarrienetakoak:

1. Egoera gabeko iragazketa.
2. Egoeradun iragazketa.
3. IP helbide eta portuen itzulpena (NAT, NAPT)[32].
4. Hedagarria hainbat moduluen bitartez.

Informazio gehiagorako (Netfilter, 2006).

4.1.6.2.- Zubiak

Gehiegi erabiltzen ez diren tresnak dira software librean oinarritutako zubiak, are eta gutxiago suhesien lana egiteko, baina software zerrenda honetan agertzea justifikatuta ikusten dut, oso tresna gomendagarria izan daitekeelako zenbait ingurunetan. OSIren bigarren mailan lan egiteak praktikan esan nahi du gure sarean ia erabat gardenak diren gailuak izango ditugula, eta hau beti da interesgarria segurtasunaren ikuspuntutik. Honekin batera, kolisio domeinuak apurtzen dituztenez, sarean gehiegizko trafikoa muga dezakete.

Informazio gehiagorako (Bridge, 2006).

4.1.6.3.- Snort

Software librearen ingurunean kokatzen bagara, eta IDS[33] baten bila hasita, berehala

32 Network Address Translation, Network Address Port Translation.

33 *Intrusion Detection System*. Onartugabeko atzipen ahaleginak detektatzen dituen sistema izan ohi da. Kontuz! Detektatzeak ez du esan nahi galerazi denik.

Snort etorriko zaigu burura. 1998.ean sortutako aplikazioa da *Snort*, heldua, beraz.

Snort sarean "entzuten" dagoen softwarea da. Erregela-base bat erabiliz ikusten duen trafikoan eraso ahalegin bat badagoen aztertzen du, eta hala detektatuko balu, abisatuko luke. Esate baterako, web konexio batean "*GET /cmd.exe*" bezalako katea ikusitakoan, "*web-iis.rules*" erregela multzoa aztertuta konexio horretan "*WEB-IIS cmd.exe access*" deskribapena duen abisua bidaliko liguke[34].

Informazio gehiagorako (Snort, 2006).

4.1.6.4.- Ntop

Askotan, sare batek denbora tarte luze xamar batez martxan dagoenean funtzionamendu patroi batzuk uzten ditu agerian. Nolabait, datu estatistikoak izango dira hauek; jakingo dugu zenbateko trafikoa izaten den, esate baterako astelehen arrunt batean, zenbat web atzipen, zenbat byte ateratzen den gure saretik, ... Datu guzti hauek ezartzen duten "normaltasun egoera" ustegabean apurtzen denean sare administratzailearen ikerketa gomendagarria izan daiteke.

Testuinguru honetan *Ntop* aukera bat izan daiteke, MRTG eta *RRDtool* softwarearekin batera, baina azken bi hauek geroxeago aipatuko ditut sakonago.

Informazio gehiagorako (Ntop, 2006).

4.1.6.5.- Argus

Datu estatistikoen bitartez lortutako informazioa erabilgarria da, noski. Jakingo dugu egun

34 Erregela zehatza hau da:
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS (msg:"WEB-IIS cmd.exe access"; flow:to_server,established; uricontent:"cmd.exe"; nocase; classtype:web-application-attack; sid:1002; rev:7;)

batean ordu konkretu batean trafikoa bikoiztu zela; baina behin ikertzen hasita, seguraski informazio gehiago beharko dugu, hala nola zenbat saio egon diren egun eta ordu horretan, zein IP helbideen artean, zein portu erabilita, TCP edo UDP izan den, zenbat trafiko pasa den saio horretatik, ...

Saioen informazioa jasotzeko aplikazioen artean aukeretako bat *Argus* da. Informazio gehiagorako (Argus, 2006).

4.1.6.6.- Tcpdump

Saioen datuetatik badakigu FTP konexio bat egon dela bi makinaren artean. Baina, zein izan da kopiatutako fitxategia? Hau ez dugu saio hutsarekin jakingo. Honetarako saio "barruan" igarotako trafikoaren kopia behar dugu, kopia osoa.

Hemen agertzen da *Tcpdump* ezaguna. Bere helburua da sarean dagoen trafiko guztia gordetzea[35], gero analisi egokia egiteko. Arestian aipatutako *Snort*-ek ere egin dezake lan hau.

Jakina, gordetako informazio hori nonbait biltegiratu behar da, eta nahiz eta biltegiratzea gero eta merkeagoa den, etengabe trafiko guztia gordetzen duen sistema bat ez da bideragarria izango ingurune guztietan.

Eta behin trafikoa gordeta nola lortu modu errazean FTP saio batean gertatutakoa? *Tcpflow* bezalako tresnak dira honetarako interesgarriak.

Informazio gehiagorako (Tcpdump, 2006) eta (Tcpflow, 2003).

35 Ez da erabat zuzena, jasotako trafikoa muga daitekeelako, baina nire ustez adierazi nahi dudan kontzeptua hobeto isladatzen da honela.

4.1.6.7.- Sguil

Tresna mordoa aipatu dut dagoeneko. Zorionez, *Sguil* bezalako aplikazioek *Snort* (alertak eta trafiko osoa gordetzeko), *Sancp* (saioetarako), *Tcpflow* eta bestelako tresnak elkartzen dituzte ingurune grafiko txukun batean.

Sguil sistema batek, laburbilduta, osagai hauek ditu:

1. Hainbat sentsore sarearen hainbat puntutan trafikoa gorde eta aztertzeko.
2. Datu-base bat, informazioa biltegiratzeko.
3. *Sguil* zerbitzaria, guztia kudeatu eta antolatzeko.
4. *Sguil* bezeroa, emaitzak grafikoki aurkezteko.

Informazio gehiagorako (Sguil, 2006).

4.1.7.- Monitorizazioa

Sarearen monitorizazioa ez da segurtasunaren ikuspuntutik bakarrik jorratu beharko dugun arloa. Sarearen mugak non dauden ikusten lagunduko digu, gaizki funtzionatzen duten gailuak errazago aurkituko ditugu, RAM memoria edo partizio batean tokia agortzen ari dela, posta zerbitzariak zenbat mezu dituen ilaratuta, datu-baseak zein eragiketa mota egiten dituen gehien,.... Zerrenda amaituezina da.

Eta ez hau bakarrik; monitorizazio-sistemek askotan grafikoak egiteko aukera ematen dute. Enpresak dituen bezeroei edo enpresaren nagusiari gauzak hitzez azaldu genkizkieke, baina kasu honetan ere irudi batek mila hitzek baino balio askoz handiago du!

4.1.7.1.-MRTG, RRDtool

Derrigorrezkoa ez bada ere, gehienetan SNMP[36] protokoloa ezkutatzen da bi tresna hauen atzetik. Protokolo hau "hitzegiten" duten aplikazio nahiz gailuetatik informazioa jaso, eta modu grafikoan aurkezten duten tresnak dira biak. Egia esan, SNMPren malgutasunari esker zenbaki bidez adierazi daitekeen edozer jarri dezakegu grafiko batean MRTG edo *RRDtool* bezalako aplikazioak erabiliz.

Informazio gehiagorako (Snmp, 1999), (Mrtg, 2006) eta (Rrdtool, 2006).

4.1.7.2.- Cacti

MRTG eta *RRDtool* tresna bikainak dira, oso erabiliak, baina konfiguratzeko astun samarrak, batez ere gailu asko monitorizatzen ditugunean. Honi konponbidea emateko hainbat tresna dago, gehienak *Apache*, PHP[37] eta *Mysql* bidez eginak. *Cacti* hauetako bat da. Ez da bakarra, baina erabilienetakoen artean dago.

Informazio gehiagorako (Cacti, 2006).

4.1.7.3.- Nagios

Nagios bezalako tresnek azpiegitura batean dauden gailu guztiak monitorizatzeko aukera ematen dute; hauen artean komunikazioetarako gailuak, web zerbitzariak, posta zerbitzariak edo datu-base kudeaketa-sistemak, gutxi batzuk aipatzearren. Zerbitzuren bat eten, edo erantzun denborak txarrak direnean ekintza bat automatikoti egiteko aukera, edo posta elektronikoz edo SMS bidez mezuak bidaltzeko modua ere ematen dute.

Informazio gehiagorako (Nagios, 2007).

36 Simple Network Management Protocol.

37 PHP: Hypertext preprocessor. http://www.php.net

4.2.- Fidagarritasunaren neurketa

Azkeneko hamarkadetan Internet eta sare korporatiboen erabilpenak sekulako garapena izan du. Gaur egun, hainbat eta hainbat enpresek sareen eta informatikaren beharra dute, askotan erabateko menpekotasuna, bai barne funtzionamendua bai bezeroekiko harremana hauen bitartez egiten baita. Etorkizunari begira ere, joerak menpekotasun hau gero eta orokortuagoa izango dela dioenez, gero eta azpiegitura sendoagoak edo egonkorragoak eskatzen dira; hitz gutxitan, fidagarritasun handikoak[38].

Interneteko atariak, erosketak egiteko web guneak edo sare korporatiboak, gutxi batzuk aipatzeko, asteko zazpi egunetan hogeita lau orduz funtzionamenduan egon behar dute. Hau bete ezean, eskainitako zerbitzuak doanekoak badira ere, ondorioak kaltegarriak izango dira. Zer esanik ez merkataritza elektronikoan diharduen web gune batean, edo sare korporatiboaren menpekotasuna duen enpresa batean. Eta larrialdietarako erabiltzen den azpiegiturak arazorik balu? Eta ospitalea balitz? Hauetan, noski, fidagarritasunaren beharrak gora egiten du.

Sistema baten fidagarritasun mailaren deskribapena gehienetan kontzeptualki egiten den zerbait da. Askotan, sare diagrama bat hartuta, trafikoaren ohiko fluxua zein den, azpiegituraren punturik ahulenek non dauden edo arazoen aurrean zer egin behar den irudien bitartez deskribatzen dira. Azpiatal honetan guzti honi formalismo minimo bat ematen ahaleginduko naiz, baina kalkulu matematikoetan gehiegi sartu gabe. Honela, nire ustez, praktikara erraztasunez eraman ahal izango dira hemen aipatutakoak. Gaian eta oinarri matematikoetan gehiago sakontzeko (Oggerino, 2001) liburua eta bertan aipatutako erreferentziak irakur daitezke.

38 Fidagarritasun handia, ingeleraz high availability irakurri ohi da.

4.2.1.- Erabilitako kontzeptuak

Fidagarritasuna neurtzen denean, eta batez ere sareen fidagarritasunari dagokionean, hiru kontzeptu nagusi erabiltzen dira. Gehienetan ingeleraz erabiltzen direnez, hemen ere honela aipatuko ditut.

- MTBF (Mean time between failure): Gailu baten hutsegiteen artean igarotzen den denbora adierazteko erabiltzen da zenbaki hau. Fabrikatzailearen esku gelditzen da balio hau ematea, normalean orduetan.
- MTTF (Mean time to failure): Gailu bat lehenengo aldiz martxan jartzen denetik hutsegitea eman arte igarotzen den denbora adierazten du. Teknikoki MTBFren berdina ez bada era, askotan sinplifikazio bat egiten da. Hemen ere ez dut balio hau erabiliko.
- MTTR (Mean time to recovery): Hutsegitea ematen denetik sistema berriz normaltasunera itzuli arte igarotako denbora. Balio honen kalkulua egiteko kontutan izan beharrekoak:
  - Detekzioa: Zenbat denbora behar da arazoa ematen denetik antzeman arte?
  - Diagnostikoa: Zenbat denbora behar da arazoa zein den jakin arte?
  - Konpontzea: Zenbat denbora igaro behar da arazoa konpondu arte?

Fidagarritasuna neurtzeko formulasinple hau erabiliko dugu:

$$fidagarritasuna = \frac{MTBF}{(MTBF + MTTR)}$$

Normalean urteko fidagarritasuna da ematen den balioa, hau da, urte batean zenbat denboraz egongo den gailu bat geldituta. Urte batean 525600 minutu daudenez, zenbaki hau erabili nezakeen kalkuluak egiteko, baina bisurtetan egun bat gehiago dagoenez, askotan gainontzeko urteetan egun laurdena gehitzen da guztiak berdin gelditzeko. Hemendik aurrera 525960 zenbakia erabiliko dut salbuespenik ez idazteko, baina norberaren esku egongo da honela egitea edo ez.

Praktikan, fidagarritasuna "saltzen" denean azken emaitzan zenbat 9 agertzen den esaten da. Askotan esan ohi da, eta hau ez da beti betetzen den erregela bat, bi 9tik aurrera berri bat lortu nahi den bakoitzean inbertsioa bikoiztu behar dela. Hurrengo taulan esandakoen adibide batzuk ikus daitezke.

Bederatzi kopurua	Fidagarritasuna ehunekotan	*Uptime*[39] minutuak (fidagarritasuna * 525960)	*Downtime* minutuak (525960 – *uptime*)	Urteko *downtime*
1	90.000%	473364	52596	35.5 egun
2	99.000%	520700.4	5259.6	3.5 egun
6	99.9999%	525959.5	0.52596	32 segundo

Taula 2: Fidagarritasunaren kalkuluaren adibideak

4.2.2.- Hutsegite eta berreskuratze denboren analisia

Kapituluaren deskribapena egin denean, kalkulu formalegirik egingo ez nuela aipatu dut. MTBF denboran, esate baterako, erraz ikusten da fabrikatzaileak emandako datuetan erabateko ziurtasunik ez dagoela. Tenperatura desegokia, gehiegizko hautsa, hezetasuna edo, besterik ez bada, gure gailuari tokatu zaiolako; emandako MTBF balioa esandakoa baino txikiagoa izan daiteke, eta oso kontutan izan beharko genuke. Hau gutxi balitz, gehienetan sistemak konplexuak dira, hardware eta softwarearen nahasketa. Nahiz eta hardwarearen MTBF balioa eduki dezakegun, nola lortu softwareari dagokiona? *Cisco*-k, adibidez, bere IOSaren inguruan zenbait balio ematen ditu, baina, nola kalkulatu aldiberean exekutatzen dauden hainbat aplikazio eta protokoloren balioa, askotan erabilpen eta guztien arteko elkarrekintzaren menpe badago? Esan bezala, askotan MTBF

39 *Uptime* da sistema funtzionamenduan dagoen denbora, eta *downtime* matxuratuta edo egoera arrunta berreskuratu gabe.

balioa formuletan aldagai moduanutzi beharko dugu.

MTTR balioa, ordea, sistemen administrazio egokiarekin hobetu ahal izango dugu. Nola gutxitu sistema bat berreskuratzeko behar dugun denbora?

- Gailuen dokumentazio ona eduki, eguneratuta. Modeloa, serie-zenbakia, software bertsioak, IP eta MAC helbideak, mantentze kontratuen informazioa....
- Ahal den neurrian mantentze kontratu onak izatea komeni da. Hogeita lau ordutako mantentze kontratu batek egun bateko MTTR balioa ematen du, gutxienez.
- Tamaina eta konplexutasun handiko saretan detekzioa eta diagnostikoa zaila izan daiteke. Ahal den neurrian azpiegitura sinple mantentzea gomendagarria da.
- Sarearen topologia, egitura, helbideratzeari buruzko informazioa edo erabilitako protokoloek arazoen aurrean portaera desberdina izan dezakete.
- Aurrekontua mugatua denean, lehentasunak jarri behar dira. Azpiegitura batean gailurik garrantzitsuenak zeintzuk diren ezagutu, eta horien gainean monitorizazio eta arazoen detekziorako sistemak ezarri.

Hauekin batera, hutsegite bat ematen denean zenbait datu jasotzea ere interesgarria izan daiteke. *Nagios* bezalako tresnak lagungarri izango zaizkigu honetarako:

- Hutsegitea erabatekoa edo partziala izanden.
- Hutsegitearen izaera, detekzioa eta diagnostikoa egin arte zenbat denbora igaro den.
- Hutsegitea noiz hasi den.
- Hutsegitea noiz bukatu den, hots, gailu, edo orokorrean, azpiegitura, noiz itzuli den ohiko egoerara. Askotan arazoa konponduta badago ere zerbitzari edo gailu desberdinek denbora behar dute egoera berreskuratzeko. Bideratze protokoloek, esate baterako, bideratze taulak berregiteko konbergentzia denbora deritzona behar dute berriz funtzionamendu egokia lortzeko.

Denboran zehar gertatutako datu guzti hauen analisiak detekzioan arazorik badagoen,

gailurik arriskutsuenak zeintzuk diren eta berreskuratze prozesuak hobetzeko aukera emango digu, sistemaren MTTR balioa txikituz.

4.2.3.- Fidagarritasunaren kalkuluaren aplikazioa

Gogoratu dezagun fidagarritasunaren kalkulurako erabiliko dugun formula:

$$fidagarritasuna = \frac{MTBF}{(MTBF + MTTR)}$$

Azken finean, esan genezake sistema bat funtzionamenduan dagoen denbora eta denbora osoaren arteko erlazio bat egiten ari garela goiko formulan. Modu honetan ere adierazi genezake:

$$fidagarritasuna = \frac{Uptime}{(Uptime + Downtime)}$$

Bi kasuetan, gure azpiegituraren fidagarritasun maila igotzeko, batetik MTBF edo sistema funtzionamenduan dagoen denbora handitu, eta bestetik sistema berreskuratzeko behar den denbora txikitu beharko genuke. Web zerbitzari batean, adibidez, MTBF balioa handitzeko kalitateko hardwarea erabiliko genuke, erredundantzia eta RAID bezalakoak erabiliz. Softwareari dagokionean, sistema eragileak eta aplikazioak ere optimizatuko genituzke, mantentze, eguneraketa edo segurtasun politika egokiekin, besteak beste. MTTR balioa txikitzeko dagoeneko zenbait ideia eman ditugu. Detekzioa hobetzeko tresna egokiak erabiltzea ezinbestekoa da, diagnostiko azkarra egiteko azpiegituraren osagaiak ezagutu behar dira, formazio egokiarekin, eta apurtutakoa konpontzeko mantentze kontratu egokiak, hardwarearen kasuan[40], eta berreskuratze prozedura eta segurtasun kopien erabilpenarekin software arazoa denean.

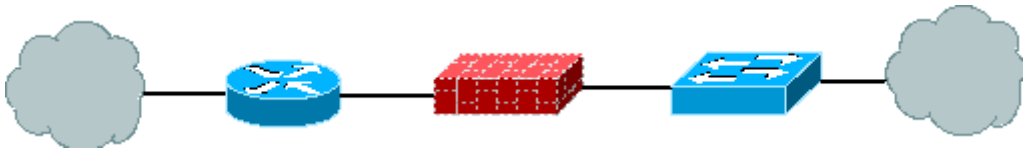
40 Software mailan ere mantentze kontratuak izan daitezke.

4.2.4.- Sare azpiegituraren fidagarritasuna

Sare azpiegitura bat diseinatzen denean gailu desberdinen kokapenak garrantzi handia du. Sistema fidagarri bat lortzeko ideia nagusia erredundantzia dela esan genezake. Honela, makina bat hondatzen denean, edo aplikazioren batean arazoren bat dagoenean, sistemaren gainontzeko osagaien artean hondatutako horren lana egingo litzateke. Arazoen antzematea eta berreskuratzea erabat automatikoa izatea lortuko bagenu, sistemak bere osotasunean fidagarritasun balio ona izango luke. Zoritxarrez, askotan ezin da erabateko erredundantzia duen azpiegiturarik lortu, aurrekontua mugatua izaten delako edo teknikoki konplexutasun handiegia dakarrelako.

4.2.4.1.- Seriean konektatutako gailuen fidagarritasuna

Hurrengo irudian seriean konektatutako gailuekin eraikitako sare bat ikusten da:



Irudia 1: Seriean konektatutako sarea

Irudian ikusten denez, bideragailua , suhesia edo *switch*-a hondatuko balira sarea erabat geldituko litzateke. Erraz ikusten da hau ez dela bilatzen den egoera. Fidagarritasunaren kalkulua azpiegitura mota honekin:

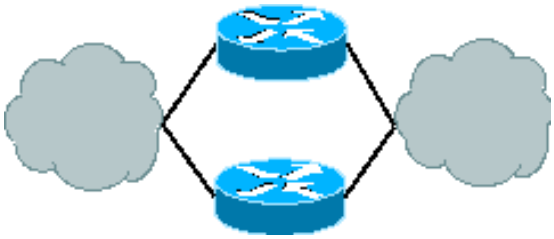
$$fidagarritasunaSeriean = \prod osagaienFidagarritasuna$$

Bideragailuaren fidagarritasuna 99.99% balitz, suhesiarena 99.93% eta *switch*-arena 99.9998%, orduan sarearena:

$$fidagarritasunaSeriean = 0.9999 * 0.9993 * 0.999998 = 0.99919$$

4.2.4.2.- Paraleloan konektatutako gailuen fidagarritasuna

Hurrengo irudian paraleloan konektatutako gailuekin osatutako sarea ikusi daiteke:



Irudia 2: Paraleloan konektatutako sarea

Kasu honetan bideragailu bat matxuratuko balitz bigarrena erabiliko litzateke. Protokolo egokiak erabilita, gainera, prozesu hau gardena eta ia denbora errealean eman daiteke. Zalantzarik gabe, honelako sistemak dira gehien interesatzen zaizkigunak, beti ere aurrekontuak ahalbideratzen duenean. Erabiltzen den formula:

$$fidagarritasunaParaleloan = 1 - [\prod (1 - osagaienFidagarritasuna)]$$

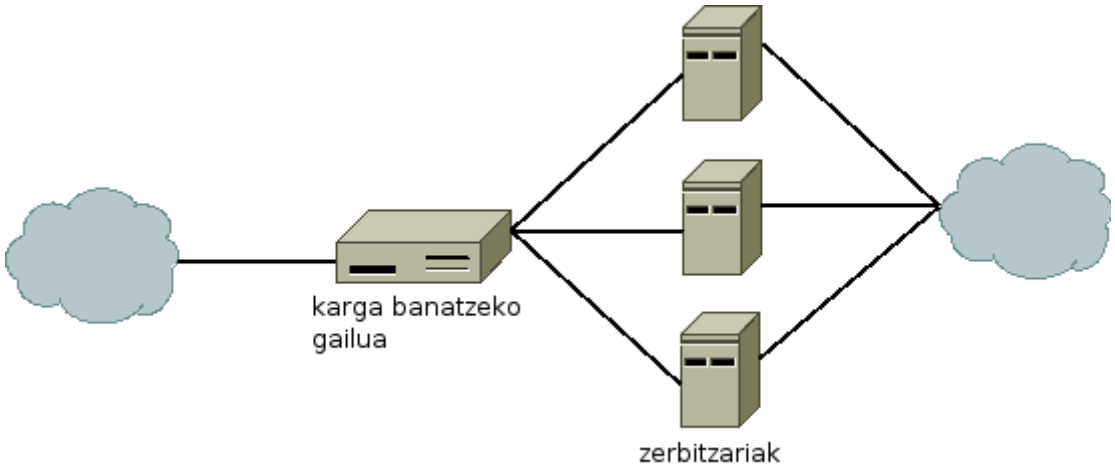
Formularen aplikazioa irudiko kasuan eta bideragailu bakoitzaren fidagarritasuna 99.99%koa balitz:

$$fidagarritasunaParaleloan = 1 - [(1 - 0.9999) * (1 - 0.9999)] = 0.99999999$$

Kontutan izan honelako diseinuek sareari erredundantzia handia gehitzen badiote ere konplexutasuna ere gehitzen diotela, eta honek eragina izan dezakeela MTBF eta MTTR balioetan. Honelako sistema bat martxan jarri aurretik gomendagarria da probak egitea, kableak askatuz, gailuak itzaliz, ...

4.2.4.3.- Serie eta paralelo nahasketak

Paraleloan erabat konektatutako sareak ez dira askotan ikusten. Gehienetan sarerik aurreratuenetan ere badago seriean konektatutako zerbait. Zenbaitetan makina guztiak bikoiztuta izatea garestiegia da (aterako zaion etekinarekin alderatuta), beste batzuetan honelako azpiegituren ezarketa eta mantentzeak eskatzen duen ordu kopurua handiegia da. Tarteko konponbide bat seriean eta paraleloan dauden gailuak nahastea da; honela azpiegituraren atalik garrantzitsuenak bikoizturik, eta garrantzi gutxiagokoak edo erraz ordezka daitezkeenak seriean jartzen dira.



Irudia 3: Serie eta paralelo nahasketa

Azpiegitura zati honetan karga banatzeko gailua erredundantziarik gabe dago, zerbitzariak, berriz, paraleloan. Honen fidagarritasunaren kalkulua zatika egiten da. Lehenengo pausoan makina bakoitzaren fidagarritasuna lortzen da. Bigarren pausoan paraleloan dauden osagaien fidagarritasuna kalkulatzen da, eta azkeneko pausoan seriean dauden osagaiena. Irudiko adibidea hartuz:

1. Karga banatzeko gailuaren fidagarritasuna: 95%
   Zerbitzari bakoitzaren fidagarritasuna: 80%

2. Paraleloan dauden osagaien fidagarritasuna:

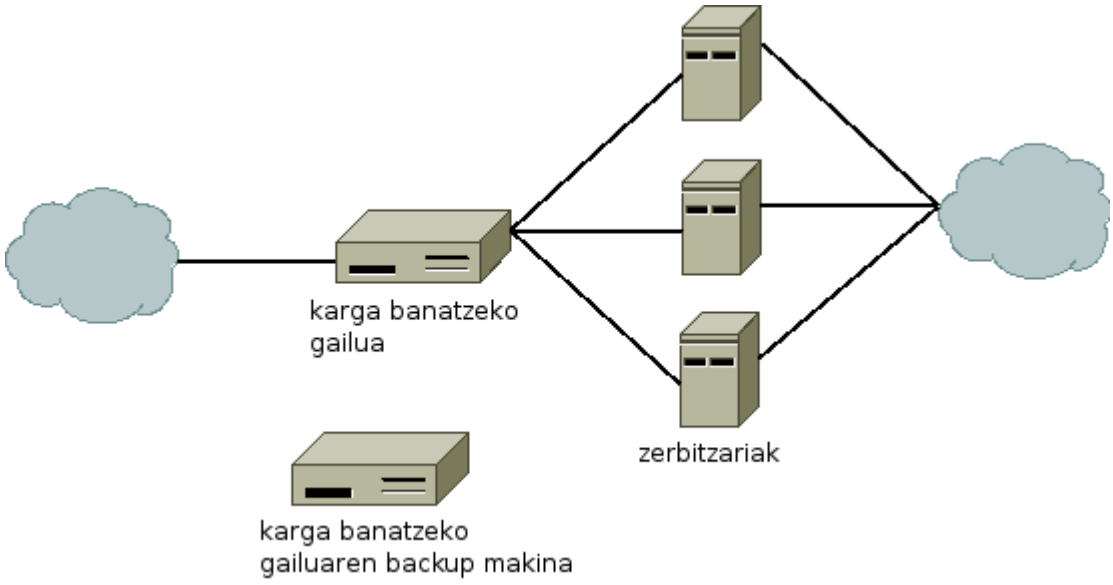
$$fidParaleloan = 1 - [(1 - 0.80) * (1 - 0.80) * (1 - 0.80)] = 0.992$$

3. Azpiegituraren fidagarritasuna

$$fidagarritasuna = 0.95 * 0.992 = 0.9424$$

4.2.4.4.- Bestelako neurriak, N + M fidagarrtasun paraleloa

Zoritxarrez, askotan sistema guztiz paraleloak garestiegiak dira. batzuetan honen arrazoia ez da gailuak ordaindu behar izatea, baizik eta honelako sareak funtzionamenduan izateko protokolo, konfigurazio eta mantentze konplexuak behar direla. Arazo honen aurrean, N + 1 (orokorrean N + M) kontzeptua erabiltzen da. Funtsean honekin esaten dena da gailu bat (+1) prest izango dugula beste baten tokia hartzeko.



Irudia 4: N + 1 fidagarritasuna

Egoera honetan karga banatzeko sistemaren kopia bat izango dugu, N + 1 beraz. Karga

banatzeko gailua matxuratuko balitz gure esku egongo litzateke *backup*-a bere ordez jartzea. Honek MTTR balioa sistema automatikoena baino askoz altuago izatea dakar, baina konfiguratzeko errazagoa da. N + 1 sistemen fidagarritasunaren kalkulua formula honekin egiten da:

$$fidagarritasuna = n * f^{(n-1)} * (1 - f) + f^n$$

f = osagai bakoitzaren fidagarritasuna (guztiena berdina dela suposatuta)
n = osagai kopurua

Kalkuluak egiten baditugu:

1. Karga banatzeko gailu bakoitzaren fidagarritasuna: 95%
   Zerbitzari bakoitzaren fidagarritasuna: 80%
2. Karga banatzeko N+1 sistemaren fidagarritasuna:

$$fidNgehi1 = 2 * 0.95^{(2-1)} * (1 - 0.95) + 0.95^2 = 0.9975$$

3. Paraleloan dauden osagaien fidagarritasuna:

$$fidParaleloan = 1 - [(1 - 0.80) * (1 - 0.80) * (1 - 0.80)] = 0.992$$

4. Azpiegituraren fidagarritasuna:

$$fidagarritasuna = 0.9975 * 0.992 = 0.9895$$

4.2.4.5.- Ondorioak

Atal honetan fidagarritasuna neurtzeko metodo baten azaleko deskribapena egin dut. Metodoa baino, kalkuluak eta lortutako balioak baino, gehiago interesatzen zait sistema baten fidagarritasunean eragina duten faktoreak ezagutzea. Guzti honekin, edozein enpresatan fidagarritasuna hobetzeko plangintza zehaztasunaz egin ahal izango dugu.

Honela, fase batean MTBF balioa hobetzeko sistema eragile edo aplikazio sendoagoak instalatzea pentsa genezake, beste batean detekziorako sistema eta berreskuratze prozedurak indartzea MTTR baliorako, beste batean *backup* makinak, edo seriean dauden gailuak paraleloan jartzea. Fase bakoitzean, gainera, hobekuntzen azalpen formala egiteko aukera izango genuke.

5.- AZPIEGITURAREN GARAPENA

Atal honen hasieran "Fnnet" ustezko enpresaren deskribapena egingo dut. Edozein aholkularitza lanaren aurreneko puntua enpresa ondo ezagutzea izan behar da. Honek esan nahi du hasteko enpresa zertan aritzen den jakin behar dela, eta ingurune horretan zergatik edo zertarako egin nahi duen inbertsioa. Helburua argi dagoenean, kontsultorearen lana izango da ideia hauek hartuta bezeroa ondo bideratzea. Honetarako, prozesua zein izango den, zenbat fasetan banatuko den, jarraituko den lan metodologia, fase bakoitzaren helburua, arriskuak, eragina enpresaren funtzionamenduan eta aurrekontua, besteak beste, behar bezala dokumentatuta egon behar dira. Proiektu honetan arlo teknikoak jorratuko ditut, baina argi izan behar da aholkularitza lan batean prozesua luzeagoa dela.

Deskribapena eginda, hurrengo azpiataletan sareko azpiegiturarekin gehien erlazionatutako puntuak landuko ditut. Hasteko bideratze eta suhesi mailak aztertuko ditut, eta gero karga banatzeko sistemekin jarraituko dut. Bukatzeko aipatutako guzti hauen ikuspegi globala eta zenbait diseinu ideia emango dut.

5.1.- Fnnet. Oinarrizko deskribapena

Fnnet liburu eta aldizkarien argitaletxe bat da[41]. Bere merkatua liburtegi nahiz liburu dendatan egon izan da beti. Orain, teknologia berrien agerpenarekin, salmenta zuzenarekin hasi nahi dute, bezeroekiko harremana hobetu eta zerbitzu gehiago emateko asmoz. Gainera, etorkizunari begira, Internet bezalako ingurunean tokia hartzea ezinbestekotzat jotzen dute.

Enpresaren arduradunen arabera, hasieran ehuneko bostean kokatuko da Internet bidez saldutakoaren kopurua. Urtero ehuneko hamar igotzeko aurreikuspena dagoenez, plangintza bat egon beharko da azpiegitura hobe dadin salmentek gora egiten duten

41 Gogoratu asmatutako enpresa dela.

heinean.

Salmenten kopuruak gora egiten duen neurrian aurrekontu gehiago izango da garapena egiteko. Ziurtatu nahi da sistema beti egongo dela funtzionamenduan, sistema gauza izango dela gabonak bezalako sasoietan bezero guztien eskaerak jasotzeko azpiegituran aparteko aldaketarik edo diru-sartzerik egin gabe, eta bezeroen datuak babesteko behar bezalako segurtasun neurriak hartuko direla.

5.2.- Hasierako azpiegitura

Fnnetek ADSL teknologia erabiltzen du bulego nagusian Interneterako sarbidea emateko. Teknologia honekin lortzen den abiadura ez da nahikoa izango zerbitzu berrirako, beraz, telekomunikazio[42] enpresa batekin harremanetan jarri eta gero, bi aukera eman dizkio; *hosting* eta *housing*[43]. Bakoitza zertan datzan hurrengo taulan ikus daiteke laburbilduta:

42 ISP (Internet Service Provider) bezala ere aipatuko dut proiektuan zehar.

43 Gehienetan ingeleraz erabiltzen dira, itzuli gabe.

	Hosting	*Housing*
Internet, atzipen mota eta abiadura	Banda zabalera bat kontratatzen da, baina bezeroa ez da arduratuko konexio motaz.	ISParen ardura da kontratatzen den banda zabalera bereroarengana eramatea *Ethernet* edo bestelakoen bitartez. Hortik aurrera bezeroaren ardura. Askotan, BGP[44] aukeran.
Hardwarearen jabetza	Alokatuta, zerbitzari batean bezero asko, edo bakarra, kontratuaren arabera. Normalean bezeroa ez da sistema eragileaz arduratzen.	Hardware guztia bezeroarena. batzuetan zenbait sare osagai alokatuta. Bezeroaren ardura da konfigurazio guztia.
Eskainitako zerbitzuak	Hardwarearen mantentzea gardena. Zerbitzariak eta aplikazioak maneiatzeko eta monitorizatzeko softwarea. Segurtasun kopiak egiteko aukera.	Aire girotu egokia. Etengabeko argindarra. Segurtasun fisikoa. Zenbaitetan segurtasun kopiak egiteko azpiegitura. Interneterako atzipena bermatua bezeroaren sarera iritsi arte.

Taula 3: Hosting eta Housing, desberdintasun nagusiak

Aukerak aztertu eta gero, azpiegitura guztiaren kontrola Fnneten esku gelditzea erabaki da, nahiz eta hasieran inbertsio handiagoa eskatu; *housing*, beraz.

Esan bezala, aukera honek egindako guztiaren gaineko erabateko kontrola emango dio Fnneti, kontratatutako argindarrarekin hasita. Instalatuko den makina kopuruaren arabera kalkuluak egin, eta toki fisikoa eta argindar beharrak aurreikusi beharko dira. Normalean edozein momentutan gehiago erosi badaiteke ere, zenbaitetan mugak ezartzen dira

44 Zenbaitetan ISPak BGPren gestio osoa egiten du, eta beste batzuetan bezeroaren esku utziko du.

tenperaturarekin arazoak egon daitezkeelako, baina ez da Fnneten kasua.

Sarbiderako *FastEthernet* interfaze bat erabiliko da. Honek esan nahi du telekomunikazio enpresak bostgarren kategoriako[45] kablea emango diola Fnneti, eta honen bitartez ISParen sarera konektatuko dela Internet atzitu ahal izateko. Hasieran, 20Mbps baino ez da kontratatuko, beraz, hor ezarriko da muga (telekomunikazio enpresaren ardura da hau). Urteekin, beharren arabera, muga hori igoko da. Inoiz 100Mbps baino gehiago beharko balira, *FastEthernet* ez litzateke nahiko izango, eta interfazea aldatu beharko litzateke. Halere, muga hori urruti dagoela pentsatzen da, eta edozein kasutan, aldaketa ez litzateke teknikoki lan gogorregia izango.

Honekin batera, telekomunikazio enpresak bere bezeroentzat eskuragarri dituen IP helbideetatik C klase[46] oso bat ere emango dio Fnneti, eta bere sarean bideratzea prestatuko du klase horri dagokion trafikoa bezeroaren ardura izango den *FastEthernet* horretara iritsi dadin. Beharrezko informazioa ondokoa da:

- ISParekin komunikatzeko sarea: 10.1.2.40/30
  - ISPak izango duen helbidea, atebide lehenetsia: 10.1.2.41
  - Bezeroak izango duen helbidea: 10.1.2.42
- Fnneti emandako C klasea: 192.168.0.0/24

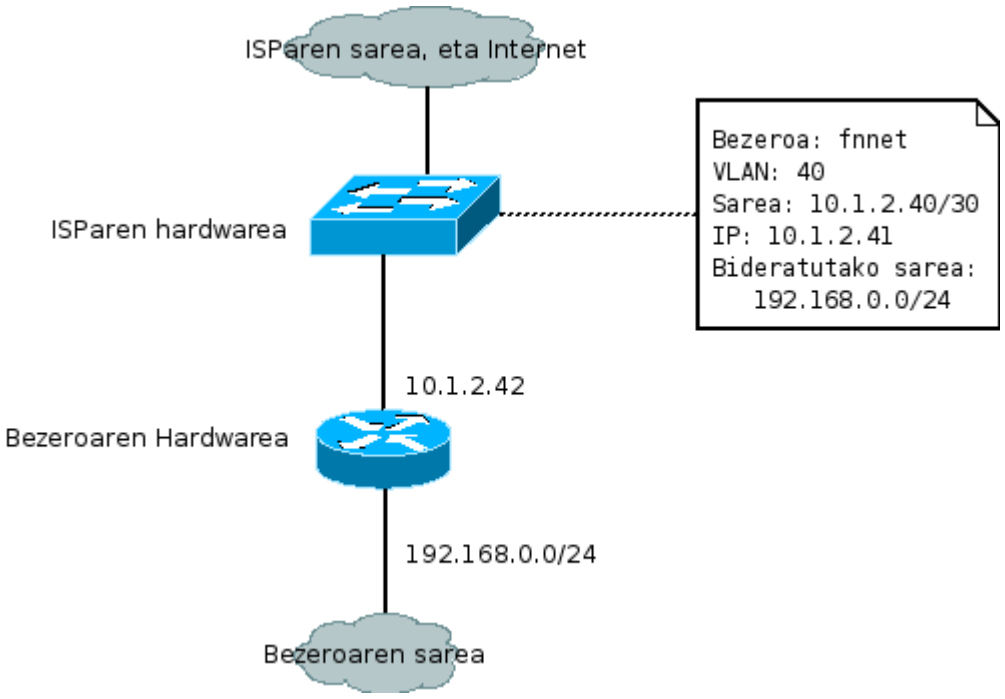
Proiektu honetan zehar helbideratze lokala erabiliko da RFC 1918[47] ezagunean adierazitako sareak erabiliz. Ingurune errealetan sare hauek ez dira Internet bidez bideragarriak, beraz, ez genituzke erabiliko. Halere, konfigurazioa berdintsua da helbideratze pribatu nahiz publikoarekin.

Emandako datu hauen adierazpen grafikoa:

45 http://en.wikipedia.org/wiki/Category_5_cable

46 http://en.wikipedia.org/wiki/Classful_network

47 ftp://ftp.rfc-editor.org/in-notes/rfc1918.txt



Irudia 5: ISPak emandako datuak grafikoki

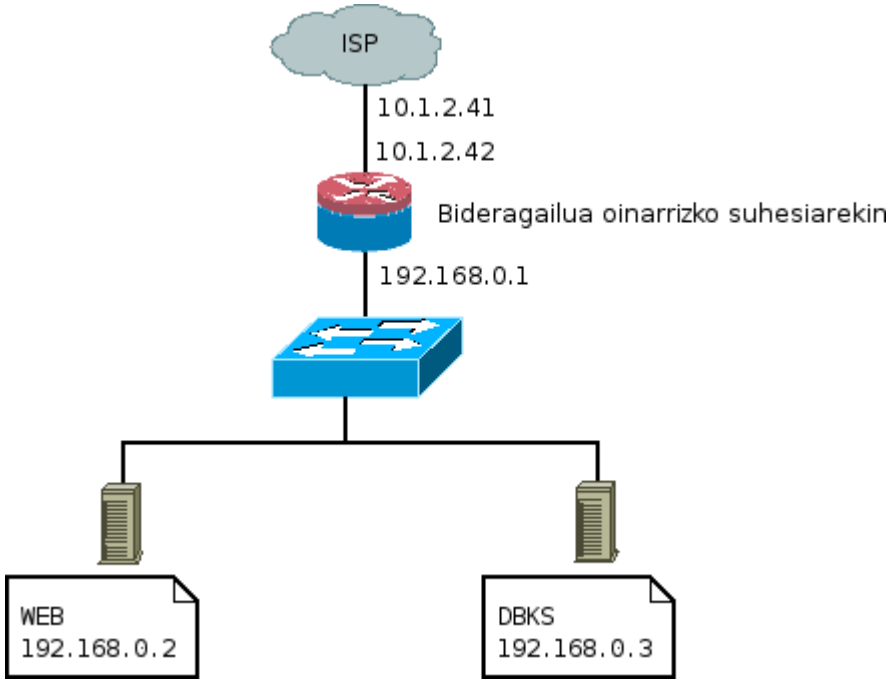
ISP enpresatan arrunta izaten da bezeroak VLAN desberdinetan sartuta egotea. Hau bezeroarentzat gardena da, baina informazioa eskuragarri badugu erabilgarria izan daiteke, batez ere goizeko bostetan arazoren bat emango balitz komenigarria izaten delako ISParen aldean dagoen teknikariari ahal diren datu gehienak ematea, nahiz eta derrigorrezkoak ez izan.

/30 maskara puntutik punturako konexioetan erabiltzen da gehienetan. Lau helbide sartzen dira azpisare mota hauetan; bat azpisarea zein den identifikatzeko, bi makinetarako eta beste bat broadcast helbidea adierazteko. Kasu honetan, ISPa eta Fnneten arteko lotura egiteko erabiliko da.

/24 maskara duen C klasea ISPak Fnneti emandako IP helbideen multzoa da. 254 helbide erabilgarri daude klase hauetan, eskainiko diren zerbitzutarako behar direnak baino askoz

gehiago. Aukera izanez gero egokia izaten da soberan izatea. Honela, urteetan sare bakarra izango dela ziurtatzen da, bideratzea eta konfigurazioa sinplifikatuz. Helbide kopuru txikiarekin, etorkizunean helbide berriak lortzeko jarraian ez dauden sare desberdinak erabili behar izatea gauza ia ziurra litzateke, eta honekin bideragailu, suhesi eta beste zenbait zerbitzari berkonfiguratu beharko lirateke, azpiegituraren zailtasuna handituz.

Hasierako sistemak web zerbitzari eta datu-base kudeaketa-sistemarako makina bana du. Aplikazioen garapena hirugarren enpresa batek egin du, *Apache* eta *Mysql*-ren gainean. ISP enpresaren komertzialari esker bideragailu eta *switch* bana utzi dira Fnneten esku hasierako instalazio eta probak egin ahal izateko. Planteatutako hasierako azpiegitura ondokoa litzateke:



Irudia 6: Oinarrizko azpiegitura

Konponbiderik sinpleena da hau, hardware minimoa erabiliz. Sistemak hasieran jasango

duen lan kargaren arabera seguraski datu-base kudeaketa-sistema eta web zerbitzaria makina berdinean sartzeko aukera egongo litzateke, baina hau ez da batere gomendagarria. Orokorrean, eta aurrekontuak ahalbideratzen duen neurrian, zerbitzariek funtzio bakarra izan beharko lukete. Hainbat arrazoi dago horretarako:

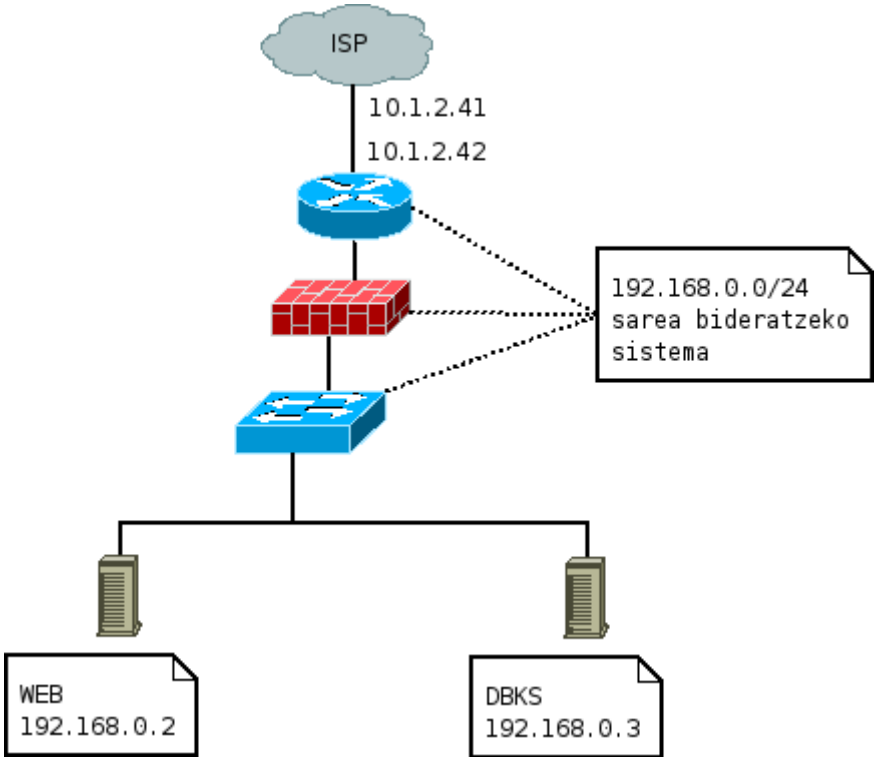
1. Makina motari dagokionean, orokorrean datu-base kudeaketa-sistemetan hardware hobeagoa erabiltzen da; fidagarriagoa, azkarragoa. DBKS gehienek memoria kopuru handia erabili dezakete datuen atzipena azkartzeko. Memoria kopuru handia nahi dugu, beraz. Disko gogor azkarrak ere behar dira memorian ez dauden datuak modu eraginkorrean jasotzeko, eta fidagarritasun handiagokoak, biltegiratzen den informazioa askotan berreskuratzeko zailagoa izaten delako. Web zerbitzarietan memoria beharra aplikazioaren araberakoa izaten da, normalean DBKS baino txikiagoa, eta disko gogorraren beharra ere arkitekturaren menpe egon ohi da. Orokorrean, edozein delarik instalatu nahi den zerbitzaria, komenigarria izaten da aurrez eskuragarri dugun hardwarean probak egitea, eta emaitzak aztertuz eta estrapolatuz behar den makina erostea.
2. Optimizazioari dagokionean ere zeharo desberdinak dira biak. Web zerbitzarietan, esate baterako, sarearekin erlazionatutako hainbat parametro aztertu beharko genituzke. Esate baterako, *net.ipv4.tcp_tw_recycle* parametroak TIME-WAIT egoeran dauden socketak azkarrago bererabiltzeko agindua ematen dio Linux kernelari[48]. DBKSetan hauek aldatzeko beharrik ez dugu izango, baina IPC arlokoak edo memoriarekin erlazionatutakoak agian bai.
3. Segurtasunaren ikuspuntutik oso desberdina da web zerbitzari publiko bat eta datu-base kudeaketa-sistema pribatu bat. Lehenengoan web trafikoa irekia izateak suposatzen du HTTP mailan segurtasun neurri bereziren bat hartu beharko litzatekeela. Bigarrenean, eta datu-base pribatua izanik, ez genuke inongo kanpo atzipenik zertan onartu. Aplikazio mailan ere datu-base kudeaketa-sistema gehienek eskaintzen dituzten segurtasunerako aukerak erabiliko genituzke.

48 Kernelaren dokumentazioan aukera guztien informazioa aurkitu daiteke.

Oinarrizko azpiegitura honetan suhesia bideragailuan bertan jarri da. Hau egokia den edo ez eztabaidagarria izan daiteke, baina proiektu honetan, eta arau orokor bezala, makina bakoitzak funtzio bat baino ez du egingo, ez bada larrialdi batetik ahal den neurrian "ihes" egiteko. Hori bai, segurtasunaz arduratutako gailu espezifikoa izateak ez du esan nahi beste makinetan segurtasun mimoa aplikatuko ez denik; batetik, suhesi nagusiaren arazoen aurrean nolabaiteko babes maila izan behar delako, eta bestetik, zenbaitetan suhesiek nekez detektatu ahal izango dituzten erasoak eman daitezkeelako. Honen adibide bat ematearren, demagun makina batean FTP zerbitzari publikoa dugula, eta IP helbide bakar batetik erabiltzaile askorekin konektatzeko ahalegin asko ematen ari dela. Demagun minutuko bi ahalegin ematen direla. Suhesiak arazorik gabe onartuko lituzke guztiak, azken finean minutuko bi baino ez dira, baina log fitxategien analisian 60 minuturen buruan 120 ahalegin ikusiko lirateke, denak okerrak. Zerbitzaria gauza balitz hau automatikoki detektatu eta neurriren bat hartzeko arazoasko ekidingo genituzke.

Bestalde, makina guztietan suhesi konplexu eta osoa jartzeko erabakia hartuko bagenu, eraginkortasun galerari mantentze arazo larria gehitu beharko genioke, gailu kopuruarekin linealki igoko bait litzateke suhesiak konfiguratzeko beharra.

Beraz, banatu ditzagun suhesi eta bideragailua:



Irudia 7: Oinarrizko azpiegituraren osagaien banaketa

Diseinu honetan suhesia eta bideragailua banatuta azaltzen dira. Itxuraz hondatu daitekeen beste gailu bat baino ez da gehitu, baina sakonago aztertuta diseinu hierarkiko eta modularrak ematen duen malgutasuna, argitasuna eta hutsegiteen isolamendua gehitu diogu sareari, garapenean zehar ikusiko den bezala. Gainera, askotan gailu gehiago egoteak ez du esan nahi azpiegitura garestiagoa izango denik. Harrigarria bada ere bi gailu sinpleekin eraikitakoa zenbaitetan gailu konplexu bat baino merkeagoa izan daiteke, are eta gehiago software librea erabiltzen badugu.

Baina, diseinu hierarkikoa alde batean utzita, oinarrizko azpiegitura hau onargarria da? Betetzen al ditu honezkero aipatutako fidagarritasun, malgutasun edo eraginkortasuna bezalako irizpideak? Zer gertatzen da zerbitzarietako bat hondatzen bada? Eta bideragailua balitz? Zer egin web zerbitzariarenahalmen maximoa gaindituko balitz?

Galdera guzti hauei erantzuna emateko formalismo bila hasita, neurtu ahal izateko unitate egokia denbora izan daiteke. Edozein arazoren aurrean, zenbat denbora emango luke zerbitzuak etenda? Orduak, egunak, eta agian, segurtasun kopiarik gabe, datuak betirako galduta. Hau onartezina da, eta eskuragarri ditugun baliabide eta aurrekontuarekin ekidin behar da. Hasieran berreskuratze prozesua eskuz egingo da, gero, aurrekontuarekin batera, automatizazioa bilatuko dugu.

Aurreneko pausoa segurtasun kopiak egiten direla ziurtatzea da. Datu-baseak, web edukia, konfigurazio fitxategiak eta interesgarria izan daitekeen guztia batetik, eta legediaren arabera zenbait log fitxategi bestetik, kopiatu beharko genituzke. Eduki guzti hau txikia den bitartean datuak bildu eta *Rsync* bidez nahi dugun tokira eramaten dituzten *script*-ak idatz ditzakegu, baina azpiegitura handitzen den neurrian modu honek sistema konplexuagoei utzi beharko die tokia, software librean edo itxian oinarrituta. Dokumentazio handia dago gaiaren inguruan; hasiera puntu bat (Backup, 2007) izan daiteke.

Hobekuntzen prozesua hasteko garapenaren planifikazioa idatzi beharko dugu. Hau egiteko faseak definituko dira. Bakoitzak helburu bat izango du, aurrekontu bat, arriskuen analisi propioa. Honela egiten badugu, modu kontrolatuan, ezusteko gutxiago izango da, ingurune berriak eskatzen duen formaziorako denbora izango dugu, eta berreskuratze prozedurak[49] eguneratuta mantentzeko arazorik ez dugu izango, besteak beste.

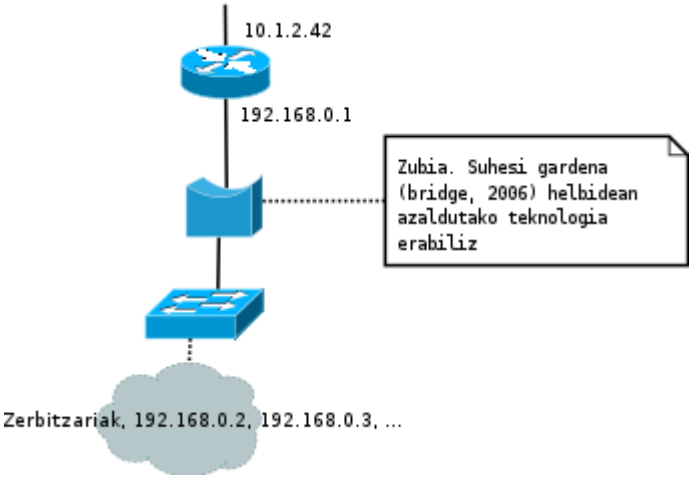
5.2.1.- Oinarrizko inplementazioa

Oinarrizko diseinua makinetara eramateko orduan zenbait erabaki ere hartu beharko dira. Esate baterako, nola inplementatuko dugu oinarrizko azpiegituran agertzen den

49 Berreskuratze prozeduretan hardware nahiz software arazoen aurrean zerbitzua berriz martxan jartzeko eman beharreko pausoak agertzen dira. Guzti hau aurrez egiaztatuta egongo da, eta ahal den neurrian sinplea izan behar da.

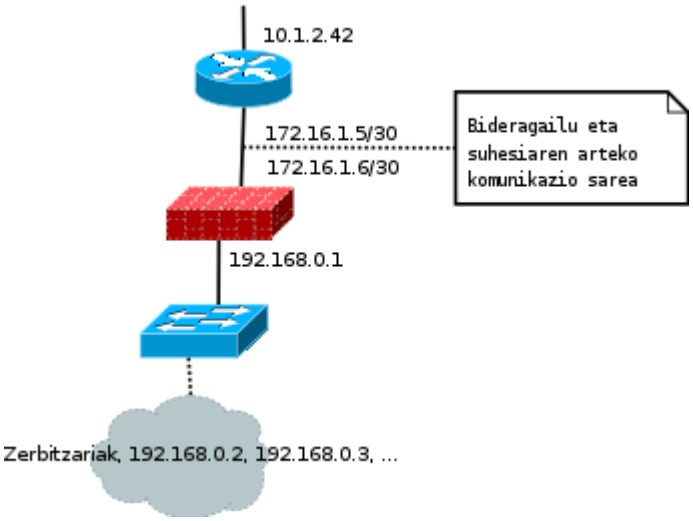
bideragailua? Kontutan izanda *Ethernet* teknologia ezaguna erabiliko dela, aukera kopuru zabala agertzen da gure aurrean. Batetik, hardware bidezko bideragailu bat eros daiteke; fabrikatzaileen katalogoa zabala da, *Cisco Systems*, *Juniper Networks*, ... Bestetik Linux edo *FreeBSD* bezalako sistema eragile libreak erabili daitezke kalitate handiko bideragailua osatzeko. Beti bezala, ez dago erabaki errazik hemen; kasu bakoitza desberdina izango da.

"Irudia 7" diagraman "192.168.0.0/24 sarea bideratzeko sistema" agertzen da. Modu desberdinetan egin daiteke honen inplementazioa. Jarraian aukeretako bi deskribatuko ditut. Ingurune errealetan bi hauetaz gain NAT suhesiak ere maiz erabiltzen dira, baina proiektu honetan bi baino ez ditut azalduko, eta bakarra erabiliko dut.



Irudia 8: Oinarrizko inplementazioa, aukera 1

Lehenengo aukera honetan zubia erabiltzen da suhesiaren lana egiteko. Gailu gardena da, hots, IP mailan dabiltzaten gailuek ez dute jakingo tartean trafikoaren iragazketa egiten dagoenik. Kontutan izan behar da OSIren bigarren mailan lan egiten duen gailua bada ere, iragazpen ahalmena ez dagoela maila honetara mugatuta, eta IP helbide edo TCP portua bezalakoak erabili ditzakeela.



Irudia 9: Oinarrizko inplementazioa, aukera 2

Bigarren honetan suhesiari IP helbide bat emango diogu. Hau egiteko /30 sare bat erabiliko dugu, publiko nahiz pribatu. Bideragailuari esango diogu 192.168.0.0/24 sarera iristeko 172.16.1.6 helbidera bidali behar duela trafikoa. Suhesiaren atebide lehenetsia 172.16.1.5 izango da. Suhesi mota honi bideratua deritzo[50]. Ez da gailu gardena, baina askotan malgutasun handiagoa eskaintzen du.

Zein aukeratu behar den ez da erantzuten erraza. Azpiegitura zertarako erabiliko den jakin behar da bi suhesi motak baliagarri diren edo ez erabakitzeko. Sare pribatu birtual asko lotu nahiko bagenitu, esate baterako, IPrik gabeko suhesi gardena ezingo genuke erabili sareen zifratzeari hasiera eta bukaerara emateko, beste gailu bat beharko genukeelarik. Haririk gabeko saretan ere zenbaitetan arazoak egon daitezke. Bestalde, suhesi gardena martxan dagoen sare batean inongo arazorik gabe ezar daiteke, inongo konfigurazio aldaketarik egin gabe, eta hau askotan oso interesgarria izan daiteke.

Puntu honetan berriz ere azpimarratu nahiko nuke proiektuan erabilitako sare guztiak lokalak direla, eta ingurune erreal batean sare hauekin lan egin badaiteke ere normalean guztietarako sare publikoak erabiliko genituzkeela.

50 Ingeleraz transparent eta routed firewall irakurri ohi da.

	Suhesi bideratua	Suhesi gardena
IP helbideen beharra	Sare interfaze bakoitzak helbide bat izan behar du azpisare desberdinetan.	Zubien funtzionamenduan ez da helbiderik erabiltzen, ez bada zubiak berak sortzen duen trafikorako edo honen mantentzerako.
Bideratzeko ahalmena	Bideratze estatikoa, OSPF edo RIP moduko protokoloak erabili daitezke beharren arabera.	Ez, ez bada mantentzerako IP helbidea adierazteko, bideratze estatikoa erabiliz.
NAT egiteko ahalmena	Bai.	Ez.
Fidagarritasuna	HSRP edo VRRP moduko protokolo ezagunak erabili daitezke inongo arazorik gabe fidagarritasuna handitzeko.	*Spanning Tree Protocol* (STP) eta bestelakoen bidez lortu ahal izango litzateke nolabaiteko fidagarritasun maila.
Suhesi ahalmena	Sistema eragilearen arabera ahalmen gehiago edo gutxiago izango du, hala nola sare birtualak, *multicast* edo zenbait trafikori lehentasuna ematea.	Zenbait muga izango ditu. Sare pribatu birtualekin, zenbait haririk gabeko sare txartelekin. Linux inplementazioa *Ethernet*-ekin bakarrik.

Taula 4: Suhesi bideratu eta gardenak, desberdintasunik nagusienak

Proiektu honetan suhesi bideratua erabiliko dut gehien bat, batez ere kontzeptu gehiago landu daitezkeelako. Jarraian orain arte diseinatu denaren zenbait inplementazio egingo ditut. Konfigurazio osorik ez dut idatziko; zubi edo sare interfazeak martxan jartzeko beharrezko diren komandoetara mugatuko naiz, bai *Cisco* IOS bai Linux erabiliz. Honen helburu nagusia software librearen bidez hardware eta software itxiak eskaintzen duen berdina egin daitekeela frogatzea da.

5.2.1.1.- Bideragailua Cisco IOS erabiliz eta Linux suhesi bideratua

Oinarrizko konfigurazio honetan bi sare interfaze dituen bideragailua erabiliko da.

```
bideragailua#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
bideragailua(config)#interface fastethernet 0/0
bideragailua(config-if)#description lotura ISParekin
bideragailua(config-if)#ip address 10.1.2.42 255.255.255.252
bideragailua(config-if)#exit
bideragailua(config)#interface fastethernet 0/1
bideragailua(config-if)#description lotura suhesiarekin
bideragailua(config-if)#ip address 172.16.1.5 255.255.255.252
bideragailua(config-if)#exit
bideragailua(config)#ip route 0.0.0.0 0.0.0.0 10.1.2.41
bideragailua(config)#ip route 192.168.0.0 255.255.255.0 172.16.1.6
bideragailua(config)#exit
%SYS-5-CONFIG_I: Configured from console by console
bideragailua#
```

Konfigurazio zati honetan ikusten denez, *Fastethernet 0/0* interfazea ISPari begira egongo da, eta *Fastethernet 0/1* suhesiari begira. Bideratze estatikoa erabiliz atebide lehenetsia 10.1.2.41 dela esango da, eta 192.168.0.0/24 sarera doan trafikoa 172.16.1.6 helbidera bidali behar dela.

Linux suhesiaren konfigurazioa bideratzeari dagokionean:

```
ip addr add 172.16.1.6/30 dev eth0
ip addr add 192.168.0.1/24 dev eth1
```

```
ip route add default via  172.16.1.5 dev eth0
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Konfigurazio hau egiteko komando orokorrak erabili ditut. Idatzitakoa edozein Linux banaketaren interfazeak hasieratzeko modura egokitzea irakurlearen esku gelditzen da. *Debian-stable* sisteman, esate baterako, */etc/network/interfaces* eta */etc/init.d/networking* fitxategiak egokitu beharko lirateke.

Suhesiaren konfigurazioa babestu nahi den azpiegituraren araberakoa da. Hurrengo kode zatian ICMP, *loopback*, SSH eta HTTP trafikoa bakarrik baimentzen duen erregela-multzoa dago (Gogoratu hau ez dela NAT suhesi bat):

```
# Defektuz ez da trafikorik onartuko.
iptables -t filter -P INPUT DROP
iptables -t filter -P FORWARD DROP

# Erregelak txukun egituratzeko kateak
iptables -t filter -N ICMP
iptables -t filter -N SSH
iptables -t filter -N HTTP
iptables -t filter -N BARNE_TRAF
iptables -t filter -N LOOPBACK

# ICMP, SSH eta HTTP trafikoa orokorrean onartu
iptables -t filter -A ICMP -j ACCEPT
iptables -t filter -A SSH -j ACCEPT
iptables -t filter -A HTTP -j ACCEPT

# 192.168.0.0/24 sareko makinetan sortutako trafikoa onartu
iptables -t filter -A BARNE_TRAF -i eth1 -s 192.168.0.0/24 -j ACCEPT
# loopback trafikoa onartu
iptables -t filter -A LOOPBACK -i lo -s 127.0.0.1 -d 127.0.0.1    \
      -j ACCEPT

# Suhesian bertan sartzen den trafikoaren arauak
iptables -t filter -A INPUT -p tcp --dport 22 -j SSH
iptables -t filter -A INPUT -p icmp -j ICMP
iptables -t filter -A INPUT -j BARNE_TRAF
iptables -t filter -A INPUT -j LOOPBACK
```

```
# Suhesian zehar igarotako trafikoaren arauak
iptables -t filter -A FORWARD -p tcp --dport 22 -j SSH
iptables -t filter -A FORWARD -p tcp --dport 80 -j HTTP
iptables -t filter -A FORWARD -p icmp -j ICMP
iptables -t filter -A FORWARD -j BARNE_TRAF
```

Suhesi bat ondo idazteko babestuko dituen aplikazioak eta sarea ezagutu behar dira. Askotan, mota guztietako ICMP trafikoa ez da baimentzen. Halaber, SSH atzipena gehienetan IP helbide zehatz batzuetatik baino ez da onartzen. Guzti hau kontutan izanda, argi dago idatzitako arauak egin daitekeenaren adibide sinplea baino ez dela.

5.2.1.2.- Bideragailua Linux erabiliz eta Linux suhesi gardena

Suhesi gardenak erabiltzen direnean 172.16.1.4/30 sarearen beharrik ez da izango. Zuzenean bideragailuak izango du zerbitzarien atebide lehenetsia izatearen ardura.

```
ip addr add 10.1.2.42/30 dev eth0
ip addr add 192.168.0.1/24 dev eth1
ip route add default via  10.1.2.41 dev eth0
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Suhesi gardenari dagokionean lan gehiago egin behar da. Batetik kernela gauza izan behar da zubiak sortzeko, eta bestetik, erabiltzaile mailako *brctl* aplikazioa beharko dugu zubien konfigurazioa egiteko. *Debian* sistema batean *bridge-utils* paketea instalatu beharko dugu aplikazio hau izateko. *Apt-get* erabiliz:

```
apt-get install bridge-utils
```

Honekin batera, esan bezala, gure sistemaren kernelak zubiak sortzeko ahalmena izan behar du. Zenbait banaketetan aurrez konpilatutako moduluak daude eskuragarri; edozein kasutan, *.config* nukleoaren konfigurazio fitxategia eskuz aldatu nahiko balitz BRIDGE_CONFIG[51] da aldatu beharreko aukera.

Zubiaren oinarrizko konfigurazioa sinplea da (kontutan izan *root@zubia:/#* Linux

51 Menuak erabiliz "Networking -> Networking options -> 802.1d Ethernet Bridging" da aukera 2.6.x kerneletan.

makinaren *prompt*-a dela, eta irteeraren zati batzuk ezabatu direla kodean lerro jauzirik ez egoteko):

```
# Konfiguratzen hasi baino lehen:
root@zubia:/# brctl show
bridge name     bridge id               STP enabled     interfaces

# IP helbiderik ez dugu nahi bi sare interfazeetan:
root@zubia:/# ip addr add 0.0.0.0 dev eth0
root@zubia:/# ip addr add 0.0.0.0 dev eth1
root@zubia:/# ip addr ls
1: lo: <LOOPBACK,UP,10000> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast ...
    link/ether 00:0c:6e:bb:f5:7d brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast ...
    link/ether 00:48:54:6c:45:ab brd ff:ff:ff:ff:ff:ff

# Zubiaren oinarrizko konfigurazioa egiteko,
# aurreneko komandoaz zubia sortzen da
# bigarren eta hirugarren komandoak zubian interfazeak gehitzeko:

root@zubia:/# brctl addbr zubia
root@zubia:/# brctl addif zubia eth0
root@zubia:/# brctl addif zubia eth1

root@zubia:/# brctl show
bridge name     bridge id               STP enabled     interfaces
zubia           8000.000c6ebbf57d       no              eth0
                                                        eth1

# Hautazkoa, zubiari IP bat emateko. Hau egiten badugu
# zubia atzigarria izango da:

root@zubia:/# ip addr add 192.168.0.254/24 dev zubia
root@zubia:/# ip link set zubia up

root@zubia:/# ip addr ls
1: lo: <LOOPBACK,UP,10000> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
```

```
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast ...
    link/ether 00:0c:6e:bb:f5:7d brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast ...
    link/ether 00:48:54:6c:45:ab brd ff:ff:ff:ff:ff:ff
4: zubia: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc noqueue
    link/ether 00:0c:6e:bb:f5:7d brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.254/24 scope global zubia
```

Sortutako zubiak *eth0* eta *eth1* interfazeak erabiltzen ditu. Ikusten denez 192.168.0.254 IP helbidea eman diogu, baina hau guztiz hautazkoa da. Zubiari segurtasunik handiena emateko helbiderik ez erabiltzea gomendatzen da. Idatzitakoa konfigurazio minimoa da, STP protokoloaren konfiguraziorik gabe.

Behin zubiarekin bukatuta, suhesia konfiguratzeko ordua da. Honetarako *ebtables* komandoa erabiliko dugu. Banaketa gehienetan izen berdineko paketean dago eskuragarri. Zubiarekin bezalaxe, kernelak ere ahalmena izan behar du[52]. Gogoratu hurrengo lerroak suhesiaren ahalmenaren adibide sinple bat baino ez dela, eta praktikara eramateko askoz gehiago landu beharko litzatekeela.

```
# Zubi erabat gardena nahi dugu, beraz helbidea kenduko diogu.
root@zubia:/# ip addr del 192.168.0.254/24 dev zubia

# Ebtables komandoen zerrenda:
ebtables -t filter -P FORWARD DROP

ebtables -t filter -N ICMP
ebtables -t filter -N SSH
ebtables -t filter -N HTTP
ebtables -t filter -N BARNE_TRAF

ebtables -t filter -A ICMP -j ACCEPT
ebtables -t filter -A SSH -j ACCEPT
ebtables -t filter -A HTTP -j ACCEPT
ebtables -t filter -A BARNE_TRAF -p ipv4 -i eth1     \
     --ip-src 192.168.0.0/24 -j ACCEPT
```

52 "Networking -> Networking options -> Network packet filtering (replaces ipchains) -> Bridge: Netfilter Configuration" da aukera nagusia 2.6.x kerneletan.

```
# arp trafikoa onartuko dugu
ebtables -t filter -A FORWARD -p arp -j ACCEPT

ebtables -t filter -A FORWARD -p ipv4 --ip-proto tcp       \
      --ip-dport 22 -j SSH
ebtables -t filter -A FORWARD -p ipv4 --ip-proto tcp       \
      --ip-dport 80 -j HTTP
ebtables -t filter -A FORWARD -p ipv4 --ip-proto icmp -j ICMP
ebtables -t filter -A FORWARD -p ipv4 -j BARNE_TRAF

# Adibide bat, demagun ziurtatu nahi dugula IP helbide bat
# MAC batean egon behar dela
# 192.168.0.26 IP helbidea duen makinak 00:16:17:D3:D7:D0 MAC helbidea
ez badu, ez onartu
# ebtables -A FORWARD -p IPv4 --ip-src 192.168.0.26        \
#     -s ! 00:16:17:D3:D7:D0 -j DROP
```

Iptables arruntarekin egindakoaren berdina egin dugu hemen, baina OSIren bigarren mailan. INPUT kateari dagozkion erregelak kendu ditut zubia ez delako atzigarria izango (IP helbiderik gabe). Kontutan izan, bederen, *ebtables* eta *iptables* konbinatu daitezkeela horrela beharko balitz.

5.3.- Bideratzearen hobekuntza. Fidagarritasuna, N+1 eta Standby sistemak

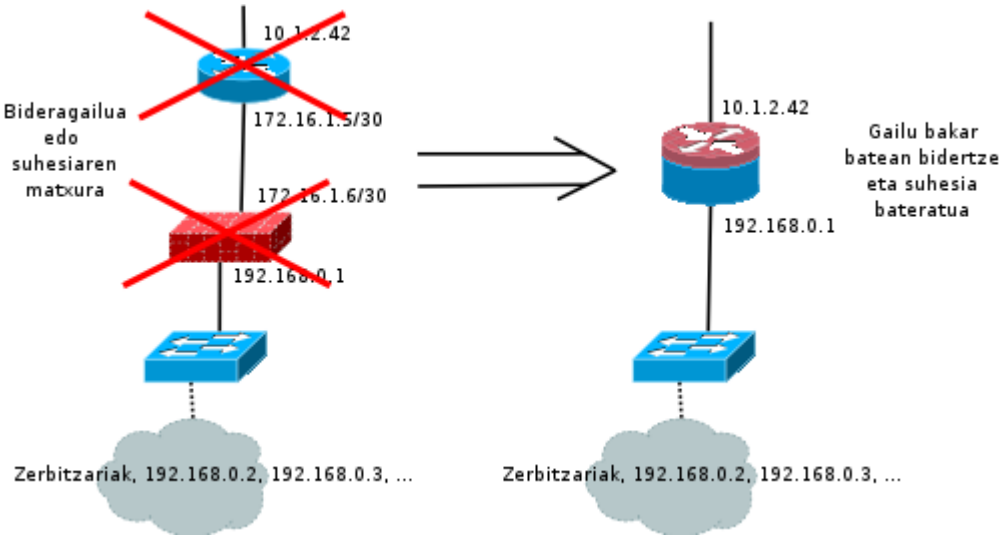
Oinarrizko diseinurako aukerak aztertu eta gero, Fnnetek Linux sistema eragilea erabiliko du bai bideragailua bai suhesia egiteko. Honela, batetik kalitate handiko sistema izango du, eta bestetik hasieran egin beharreko inbertsioa txikia izango da. “Irudia 9” topologian ikusitako diseinua inplementatuko da suhesi bideratua erabiliz.

Azpiegitura puntu honetan dagoen bezala hutsunez beteta agertzen da. Arazo larriak izan ditzakegu (denboran neurtuta) sareko edozein gailu hondatzen bada. Bideragailu, suhesi, web edo datu-base zerbitzariak, hauetako edozeinek hardware nahiz software arazorik balu, eta aurrez ondo finkatutako berreskuratze prozedurarik gabe, ordutako etenaldia ia

ziurtatuta legoke. Zorionez, oinarrizkoa bada ere, azpiegiturak bideak eskaintzen ditu fidagarritasuna hobetzeko. Bideratze mailarekin hasiko naiz.

5.3.1.- Bideragailu eta suhesiaren nahasketa

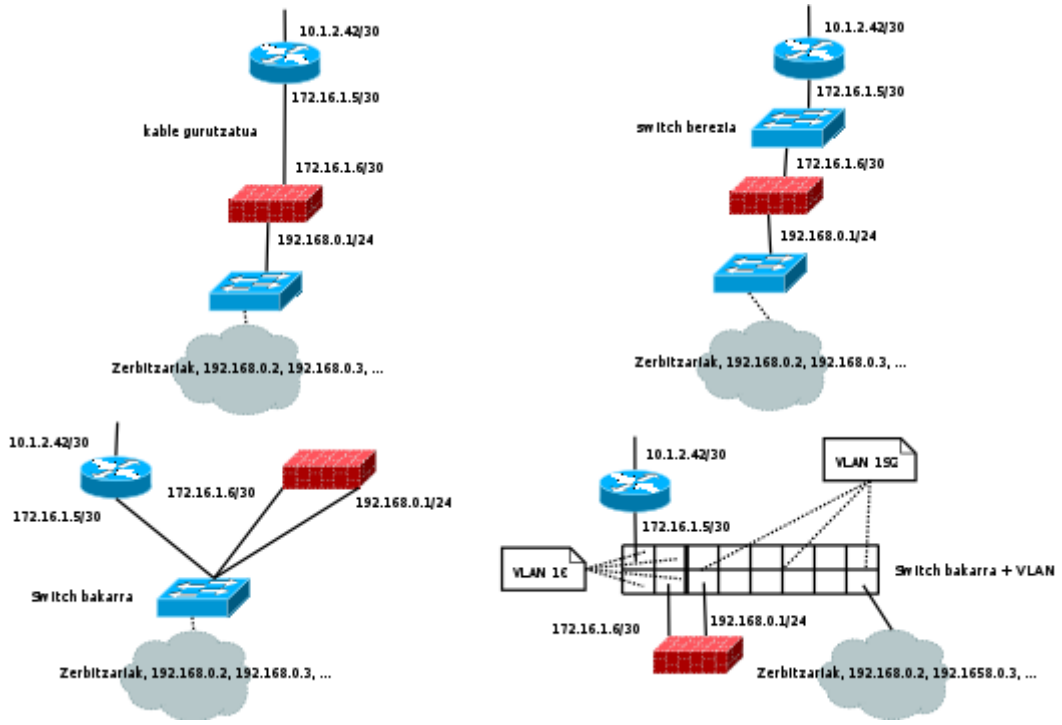
Oinarrizko azpiegituraren inplementazioan ikusten zenez, bideragailu eta suhesiaren artean, funtsean, aparteko desberdintasunik ez dago. Makina bakoitzak funtzio nagusi bakarra izan behar duela oinarrizko arau bezala aipatu dut, baina momentu honetan, eta eskuragarri dagoen hardwarea kontutan izanda, bideragailu eta suhesia derrigorrez izan behar dira bata bestearen lana egiteko gauza. Honela, bideragailua apurtuz gero suhesiak hartuko luke bideragailu eta suhesiaren lana egitearen ardura. Modu berean, suhesia hondatuko balitz bideragailuak hartuko luke aurrekoaren tokia. Fidagarritasunaren ikuspuntutik seriean dauden bi gailu nolabait paraleloan jarriko ditugu honekin. Ez da erabateko paralelotasuna izango, kableak eskuz aldatu eta zenbait *script* exekutatzeko denbora behar delako, baina seriean lortutako denborak hobetzen dira, zalantzarik gabe. Kontzeptualki, irudi bidez, honelako zerbait lortu nahi dugu:



Irudia 10: Bideragailu eta suhesia bateratuta

Oraingoz, eta garapen faseak gehiegi ez nahasteko asmoarekin, ISPak ematen digun kablea zuzenean bideragailura doala suposatuko dut. Irudian agertzen den bideragailu eta suhesiaren arteko konexioa egiteko moduari dagokionean, gutxienez lau modu desberdinetan egin daitekeela esan daiteke: Batetik kable gurutzatuarekin, bigarren aukera *switch* bat jarriz bien artean, hirugarren aukera dagoeneko azpiegituran dagoen *switch* berdina erabiliz makina guztietarako; eta azkenik, *switch* bakarrean VLAN[53] desberdinak konfiguratuz.

53 VLAN bidez broadcast domeinuak banatzen dira. Switch berdineko VLAN desberdinetan dauden gailuak ez dira defektuz elkarren artean komunikatu ahal izango, ez bada bideragailu bat erabiliz.



Irudia 11: Bideragailu eta switch arteko konexio motak

Esan bezala, oraingoz ISParen kablea zuzenean joango da bideragailura. Hau honela, eman litezkeen arazoen aurrean eskuz aldatu beharko litzateke konexioa. Honek seguraski berreskuratze denbora luzatuko luke, batez ere arazoa lan orduetatik kanpo emango balitz, detekzioa normalean motelagoa izaten delako. Hala eta guztiz ere, ISParekin dagoen loturan fidagarritasuna lortzeak azpiegitura/aurrekontu handi xamarra eskatzen duenez, askotan galdutako denbora onargarritzat hartzen da.

	Aldekoak	Kontrakoak
Kable gurutzatua	Sinplea eta merkea.	Malgutasunik ez. Arazoen aurrean berreskuratze denbora luzeagoa.
Switch berezia	Malgutasun handia. Bi *switch* sarean, beraz, larrialdietarako *backup* aukera.	*Switch* berria erosteko beharra.
Switch bakarra	Sinplea, inbertsio berririk gabe.	Zerbitzarien *switch*-ean iragazketarik gabe trafiko guztia. Segurtasunaren beharra.
VLAN desberdinak	Malgutasun handia. Inbertsio berririk ez.	*Switch*-ek VLANak egiteko ahalmena behar dute. Konfigurazioa konplexuagoa da.

Taula 5: Bideragailu eta suhesiaren arteko konexioa

Garapen fase honetan eskuragarri dagoen aurrekontuarekin *switch* bakarrean VLAN desberdinak konfiguratzea da Fnneti aurkeztu zaion aukera. Jarraian, bai bideragailua, bai suhesiaren matxuraren aurrean egin beharrekoen berreskuratze prozedura deritzona aurkeztu behar zaio enpresari.

5.3.1.1.- Oinarrizko berreskuratze prozedura

Berreskuratze prozedura batean edozein software nahiz hardware arazoren aurrean hartu beharreko ekintzen zerrenda agertu behar da. Pauso hauek aurrez probatuta egon behar dira, gainera, ahal den neurrian sinpleak eta azkarrak izan beharko lirateke. Esate baterako, oso egokia izan daiteke gailu bakoitzeko arazo posibleen deskribapena eta konponbidearen zerrenda idaztea.

1. Arazoa: Bideragailuak ez du erantzuten.
    1. Pausoa: Bideragailua berrasieratu teklatu eta monitorearekin. Software arazoa bada seguraski honekin egoera arruntera itzuliko da. Bestela, edo hardware arazoa bada, irakurri bigarren puntua.
2. Arazoa: Bideragailuak hardwarearazoa du edo berrasieratzea alferrikakoa.
    1. Pausoa: ISPtik bideragailura doan kablea zuzenean *switch*-ean jarri[54]. Erabili beharreko VLAN zenbakia 10 da, beraz, aukeratu honetan libre dagoen portu bat, adibidez, bigarrena.
    2. Pausoa: Itzali bideragailua.
    3. Pausoa: Suhesiaren *eth0* sare-txartelari bideragailuari zegokion 10.1.2.42/30 helbidea esleitu behar zaio. Suhesian *script* txiki bat dago hau automatikoki egiteko. Honek funtzionatuko ez balu, eskuz exekutatu:

```
ip addr del 172.16.1.6/30 dev eth0
ip addr add 10.1.2.42/30 dev eth0
```

    4. Pausoa: Egiaztatu zerbitzariak atzigarriak direla.
3. Arazoa: Suhesiak ez du erantzuten.
    1. Pausoa: Suhesia berrasieratu teklatu eta monitorearekin. Software arazoa bada seguraski honekin egoera arruntera itzuliko da. Bestela, edo hardware arazoa bada, irakurri laugarren puntua.
4. Arazoa: Suhesiak hardware arazoa du edo berrasieratzea alferrikakoa.
    1. Bideragailutik *switch*-era doan kablea (*eth1* interfazea) 192. VLANean jarri,

54 Fnneten *switch*-a gauza da kablea automatikoki "gurutzatzeko". "auto mdix" da honetarako agindua.

libre dagoen portu batean, adibidez, bostgarrenean.

2. Pausoa: Itzali suhesia.
3. Pausoa: Bideragailuaren *eth1* sare-txartelari suhesiari zegokion 192.168.0.1/24 helbidea esleitu behar zaio. Bideragailuan *script* txiki bat dago hau automatikoki egiteko. Honek funtzionatuko ez balu, eskuz exekutatu:

```
ip addr del 172.16.1.5/30 dev eth1
ip addr add 192.168.0.1/24 dev eth1
```

4. Pausoa: Egiaztatu zerbitzariak atzigarriak direla.
5. Pausoa: Egiaztatu zerbitzarietan oinarrizko suhesia aktibatuta dagoela.

5. Arazoa: *Switch*-ak ez du erantzuten.
   1. Pausoa: Berrasieratu, eta ikusi egoera arruntera itzultzen den. Ez bada honela gertatzen, ez dago *backup*-erako hardwarerik, beraz, ISPari bat eskatu. Azaldu denbora txiki baterako dela, eta Fnneteko administrazio departamentua jakinaren gainean jarri alokairua ordaindu ahal izateko.

Prozedura multzo honek ohiko egoera arrunta hautsi egin du. Gomendagarria da, behin arazoa konponduta, berriz egoera berreskuratzea, beti ere aurretik planifikazio egokia eginaz. Gogoan izan, bestalde, prozedura honetan hainbat *script* dagoela aipatzen dela. Zalantzarik gabe, hauek behar bezala dokumentatuta egon beharko lirateke.

5.3.2.- N+1 eta Standby sistemak

Standby hitza oso erabilia da sarearen fidagarritasunaz ari garenean. Funtsean, zain dagoen gailu bat baino ez da, bideragailu, suhesi, web zerbitzari edo beste edozein. Noski, zain dagoen web zerbitzariak momentuan lanean dagoenaren konfigurazio berdina izan behar du, honelako sistemen helburu nagusia arazoen aurrean automatikoki bestearen tokia hartzea baita. Esan genezake *Standby* sistemekin N+1, edo orokorrean N+M, fidagarritasun neurriak inplementatzen ditugula.

Zenbaitetan, *Standby* sistemak *Hot-standby* eta *Cold-standby* bezala sailkatzen dira. *Hot-*

standby moduan dagoen gailu bat VRRP edo HSRP bezalako protokoloen bitartez edozein momentutan IP helbide bat bere gain hartzeko prest dago, ordurarte funtzionamenduan egon denaren tokia berehala hartu ahal izateko. *Cold-standby* moduan dagoena itzalita dago, baina beharrezkoa balitz, piztu eta lanean hasteko prestaturik, bestelako inongo konfigurazio beharrik gabe.

Standby sistemak oso interesgarriak dira software librearen textuinguruan. Enpresa gehienetan erabiltzen ez diren konputagailu zaharrak daude. Linux bezalako sistema eragileei esker aparteko inbertsiorik gabe gure sarearen fidagarritasuna asko handitu dezakegu gailu hauek erabilita. Eraginkortasunean galera txikia egon daiteke, baina hau beti izango da ezer ez zerbitzatzea baino hobeagoa.

Proiektuan zehar *Standby* sistemen kontzeptua VRRP eta *Keepalived* softwarearen bitartez inplementatuko da Fnneten azpiegituran.

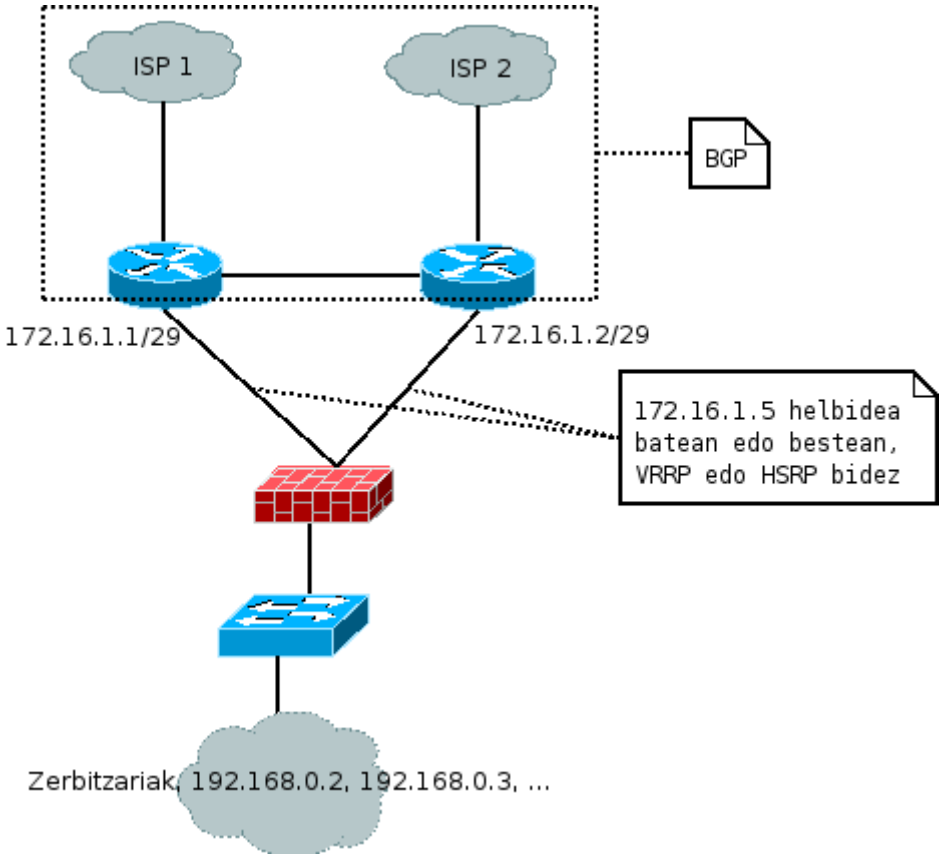
5.4.- Bideratzearen hobekuntza. HSRP/VRRP eta BGP

ISPak eskainitako zerbitzuen deskribapenean gure bideragailua eta Internet arteko komunikazioa bermatua dagoela esan da. Berme hau kontratuetan 99.9%, 99.99% edo 99.999% itxurako balioekin aurkeztu ohi da. Ez da, beraz, erabateko fidagarritasunik ematen. Honen aurrean bi aukera nagusi ditugu:

1. Onartu denbora tarte txiki batez zerbitzurik gabe egon gaitezkeela. Aurrekontua apala den bitartean hau onargarria izan daiteke, baina denborarekin Fnnetek esan beharko du ea hau ondo dagoen edo ez.
2. ISP bakarrarekin dugun menpekotasuna apurtu. Honetarako, zenbait enpresek banda-zabalera hornitzaile anitzekin dituzte akordioak, eta eraikin berdinean hauetako edozein aukeratzeko modua ematen dute zerbitzariak tokiz mugitu gabe.

Bigarren aukerak inbertsio handia eskatzen du. Batetik bi banda zabalera hornitzaileei

ordaindu beharko zaie, bestetik, hardware berria behar da, eta azkenik, mantentzea ere zailtzen da. Grafikoki, kontzeptualki, honelako zerbait bilatzen da:



Irudia 12: BGP eta HSRP/VRRP

Honen inplementazio eta mantentzerako inbertsioa, esan bezala, izugarri handitzen da. Horretarako ezagutu behar den BGP protokoloaren inguruan milaka lerro dago idatzita, baina hogei orrialdetako sarrera (Lucas, 2004) liburuxkan aurki daiteke.

Irudian HSRP/VRRP protokoloak ere agertzen dira. Kontutan izan HSRP *Cisco Systems* enpresaren protokoloa dela, eta bere gailuetan baino ezin dela erabili. VRRP, ordea, bertsio estandarra da, eta bai *Cisco* gailuetan bai gainontzekoetan dago inplementatuta. Edozein dela ere aukeratutakoa, 172.16.1.5 helbidea birtuala izango da, ez da interfaze

fisiko bati atxikia egongo. Bi bideragailuen interfazeetan 172.16.1.1/29 eta 172.16.1.2/29 helbideak[55] egongo dira, eta 172.16.1.5/29 bietako batean, bakarrean. Honen ardura duen bideragailua hondatuko balitz, besteak jasoko luke helbide birtualaren jabetza. Hau suhesiaren ikuspuntutik erabat gardena da (gogoratu 172.16.1.6/29 helbidea duela, eta 172.16.1.5rekin komunitzen dela).

BGPren konfigurazioan sartuko ez banaiz ere, zenbait kontzeptu azaldu daitezke puntu honetan. Hasteko, kontutan izan behar da BGP ez dela berez karga banatzeko ahalmena duen protokoloa, nahiz eta zenbaitetan simula daitekeen. Bestalde, bi ISPak erabili ahal izateko gure bideragailuen artean ere BGP erabili beharko genuke. Honela, momentuan 172.16.1.5 helbidearen jabetza duen bideragailuak BGP bidez eraikitako bideratze-taulak erabiliz jakingo luke zein ISP litzatekeen egokiena Interneteko helburu batera iristeko. ISP batekiko konexioa apurtuko balitz, sistemak berak detektatuko luke, eta trafiko guztia funtzionamenduan legokeen beste hornitzailera bideratuta izango litzateke. Ondorioz, gure azpiegiturari begira VRRP/HSRPk emango digu fidagarritasuna, eta Internetera begira BGP.

5.4.1.- HSRP Cisco IOS erabiliz

HSRP protokoloa itxia denez, Fnneterako eraikitako Linux bideragailu eta suhesietan ez dugu erabiliko. Halere, eta berdina egiteko bi modu erakusteko asmoz, atal honetan *Cisco* bideragailu batean HSRPren konfigurazioa zein izango litzatekeen idatziko dut. Hurrengo ataletan *Standby* sistemekin jarraitu dut, hemen agertutako kontzeptu berdina VRRP bidez inplementatuz.

HSRPren oinarrizko konfigurazioa oso sinplea da. Batetik bi bideragailuen interfazeetan 172.16.1.1/29 eta 172.16.1.2/29 helbideak konfiguratuko ditugu, eta bietako batean 172.16.1.5/29 helbide birtuala.

55 /30 maskaran 2 helbide baino ez dira onartzen interfazeetarako. Orain lau behar direnez, 172.16.1.1, 172.16.1.2, 172.16.1.5 eta 172.16.1.6, gutxienez /29 maskara bat behar da.

```
bideragailua1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
bideragailua1(config)#interface fastethernet 0/1
bideragailua1(config-if)#description lotura suhesiarekin
bideragailua1(config-if)#ip address 172.16.1.1 255.255.255.248
bideragailua1(config-if)#standby 1 ip 172.16.1.5
bideragailua1(config-if)#exit
bideragailua1(config)#exit
%SYS-5-CONFIG_I: Configured from console by console
bideragailua1#
```

Modu berdinean egingo litzateke bigarren bideragailuan ere:

```
bideragailua2#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
bideragailua2(config)#interface fastethernet 0/1
bideragailua2(config-if)#description lotura suhesiarekin
bideragailua2(config-if)#ip address 172.16.1.2 255.255.255.248
bideragailua2(config-if)#standby 1 ip 172.16.1.5
bideragailua2(config-if)#exit
bideragailua2(config)#exit
%SYS-5-CONFIG_I: Configured from console by console
bideragailua2#
```

HSRP bidez kontrolatu nahi ditugun interfazeetan *Standby* helbidea ezarri dugu. Bideragailuek erabakiko dute zeinek izango duen helbidearen kontrola. Noski, konfigurazio aurreratuagoak ere egin daitezke, baina funtsean idatzitakoarekin nahikoa da sistema martxan jartzeko.

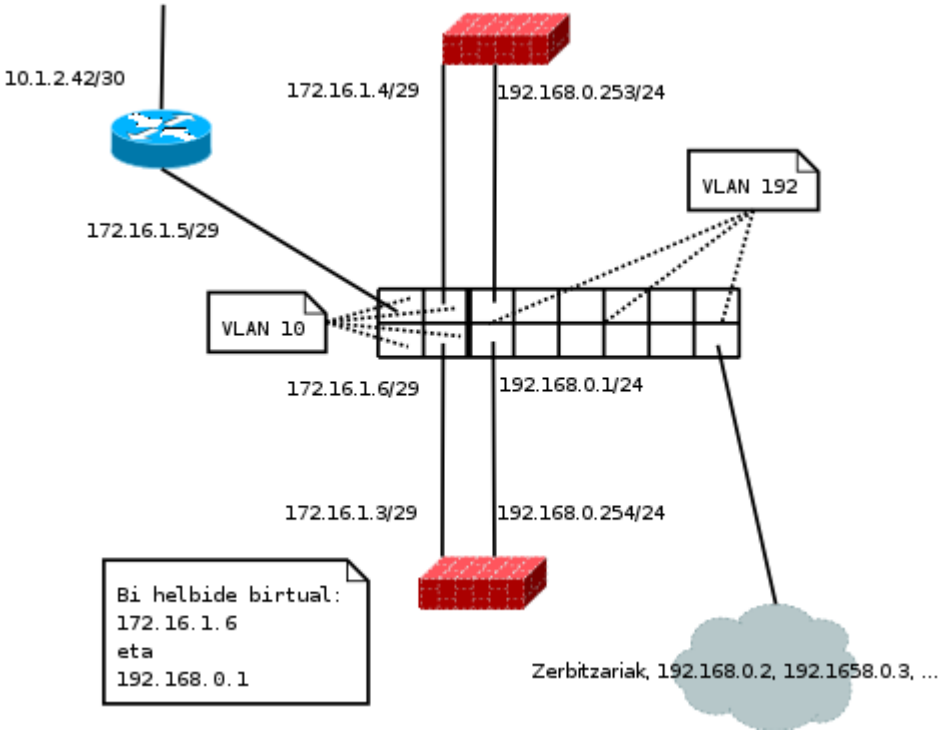
5.5.- Suhesien hobekuntza. VRRP eta N+1

Bideratze sistemaren fidagarritasuna handitzen aritu gara orain arte. Erabilitako kontzeptuak, ordea, ez dira soilik maila honetarako aplikagarriak. *Standby* sistemak berdin erabil daitezke suhesiei dagokienean ere.

BGPri buruz aipatutakoak albo batera utzita, aurrekontu beharrak ere oraingoz minimoan mantendu ditugu. Planteatutako aukera guztiak software librearen bitartez

inplementagarriak dira, eta mantentzeari dagokionean ere aparteko lanik eskatzen ez dutenak. Suhesi sistemaren hobekuntzak ere joera hau jarraituko du.

Kontzeptualki, konfigurazio berdineko bi suhesi eduki nahi ditugu. Bideragailuekin bezalaxe, helbide birtual bat mantenduko dute zerbitzarien atebide lehenetsiaren papera egiteko. Grafikoki ondokoa bilatzen dugu:



Irudia 13: Suhesiak helbide birtualekin

Irudia argiagoa izateko asmoz sinplifikazio bat egin dut, bideragailu bakarra jarriz. Suhesietan bi helbide birtual jarri ditut. Bat zerbitzariei begira, eta bestea bideragailuei begira. Honela, bideragailu eta suhesiaren artean 172.16.1.5 eta 172.16.1.6 helbideak[56]

56 /29 maskarak 6 helbide erabilgarri onartzen ditu, bina bideragailu eta suhesietarako, eta beste bi helbide birtualetarako.

erabiliko dira, eta 192.168.0.1, zerbitzariei begira. Garrantzitsua da berriz ere azpimarratzea guzti hau sareko beste gailuentzat erabat gardena dela.

Keepalived softwarearen bitartez irudian agertutako hau Linux makinetara eramango dugu. *Hot-Standby* sistema izango da, baina momentu honetan, eta inplementazioarekin jarraitu baino lehen argi geratu behar da zenbait ingurunetan *Cold-Standby* ere guztiz baliozkoa izan daitekeela *backup*-erako makinak piztu arte eman daitekeen etenaldia onargarria bada. Gure betebeharra izango da Fnneti jakinaraztea aukera hau ere baduela.

5.5.1.- Oinarrizko inplementazioa

Keepalived softwarea (Keepalived, 2006) web gunetik eskuratu daiteke. Konpilatzeko prozesua oso sinplea da, jaitsitako *keepalived-1.1.13.tar.gz* fitxategian bertan beharrezko informazioa guztia dago.

Hasteko, gogoratu ditzagun bi suhesietan egongo diren interfazeak. Lehenengo suhesian:

```
ip addr add 172.16.1.4/29 dev eth0
ip addr add 192.168.0.253/24 dev eth1
ip route add default via 172.16.1.5 dev eth0
```

Eta bigarrenean:

```
ip addr add 172.16.1.3/29 dev eth0
ip addr add 192.168.0.254/24 dev eth1
ip route add default via 172.16.1.5 dev eth0
```

Behin hau eginda *Keepalived* konfiguratzeko ordua da. Konfigurazioa */etc/keepalived/keepalived.conf* fitxategian egiten da. Lehenengo suhesian:

```
global_defs {
   notification_email {
     abisuak@posta_domeinua
   }
   notification_email_from abisuak@posta_domeinua
   smtp_server posta_zerbitzaria_ip
```

```
   smtp_connect_timeout 30
   router_id LVS_DEVEL
}

vrrp_sync_group VG1 {
    group {
      VI_1
      VI_2
    }
}

vrrp_instance VI_1 {
    state MASTER
    interface eth0
    smtp_alert
    virtual_router_id 51
    priority 150
    advert_int 1
    virtual_ipaddress {
        172.16.1.6
    }
}

vrrp_instance VI_2 {
    state MASTER
    interface eth1
    smtp_alert
    virtual_router_id 52
    priority 150
    advert_int 1
    virtual_ipaddress {
        192.168.0.1
    }
}
```

Eta bigarrenean:

```
global_defs {
   notification_email {
     abisuak@posta_domeinua
   }
   notification_email_from abisuak@posta_domeinua
   smtp_server posta_zerbitzaria_ip
```

```
   smtp_connect_timeout 30
   router_id LVS_DEVEL
}

vrrp_sync_group VG1 {
    group {
      VI_1
      VI_2
    }
}

vrrp_instance VI_1 {
    state BACKUP
    interface eth0
    smtp_alert
    virtual_router_id 51
    priority 100
    advert_int 1
    virtual_ipaddress {
        172.16.1.6
    }
}

vrrp_instance VI_2 {
    state BACKUP
    interface eth1
    smtp_alert
    virtual_router_id 52
    priority 100
    advert_int 1
    virtual_ipaddress {
        192.168.0.1
    }
}
```

vrrp_sync_group aginduarekin adierazi nahi dugu 172.16.1.6 eta 192.168.0.1 helbideak egoera berdinean egon behar direla. *vrrp_sync_group* agindurik gabe, 172.16.1.6 IParen kablea askatuko balitz beste makinak hartuko luke helbidearen jabetza, baina 192.168.0.1 aurreneko makinan jarraituko luke. Argitasunagatik eta arazorik ez izateko bi helbide birtualak makina berdinean izango ditugu, beti.

Sistema funtzionamenduan jartzeko minimoa da hau. Oinarri honetatik aurrera konfigurazio askoz konplexuagoak egin daitezke. Modu berean, fitxategi hauetan bertan karga banatzeko sistema kontrolatzeko atala gehituko dugu.

Guzti hau egin dugunean *Keepalived* deabruak exekutatuta lehenengo suhesiak hartuko luke bi IP helbide birtualen jabetza. Kableak kenduz, jarriz, makinak piztuz, itzaliz, ... helbideak makina batetik bestera mugitzen ikusiko genituzke.

5.6.- Karga banatzeko sistemak

Egindako hasierako bileretan, orain arte landutako fidagarritasun eta segurtasunari behar bezalako garrantzia eman zitzaien, baina Fnnetek bestelako ardura mota bat ere badu. Bezeroen kopuruak gora egiten duen neurrian, eskainitako zerbitzuak handitu ahala, gero eta ahalmen gehiago beharko da; baina inbertsioaren planifikazioan ez da hasieratik behar baino diru gehiago jarri nahi. Beste hitz batzuetan, ez da ahalmen osoaren ehuneko hogeian baino ez dagoen azpiegitura bat ordaindu nahi. Oinarrizko planteamendua da sistema hedatuko dela ahalmenaren hirurogeita hamarrera iritsitakoan, eta hau egiteko zerbitzuan inongo etenaldirik ez litzatekela nabaritu beharko.

Galdera guzti hauen erantzunean karga banatzeko sistemak daude. Hauei esker, web edukia[57] zerbitzatzearen ardura makina bakarrean izan ordez, hainbat zerbitzariren artean banatuko dugu. Hauetako bat matxuratuko balitz, bere lanaren zatia beste guztien artean burutuko litzateke, eta modu berean, zerbitzari berriak jarri ahalko genituzke inongo zerbitzu etenaldirik gabe.

5.6.1.- Karga banatzeko sistemen oinarrizko arkitektura

Karga banatzeko sistema batek hainbat kontzeptu berri plazaratzen ditu:

57 Berdina aplikagarria da FTP, posta elektroniko, DNS, Cache eta beste hainbat zerbitzutarako.

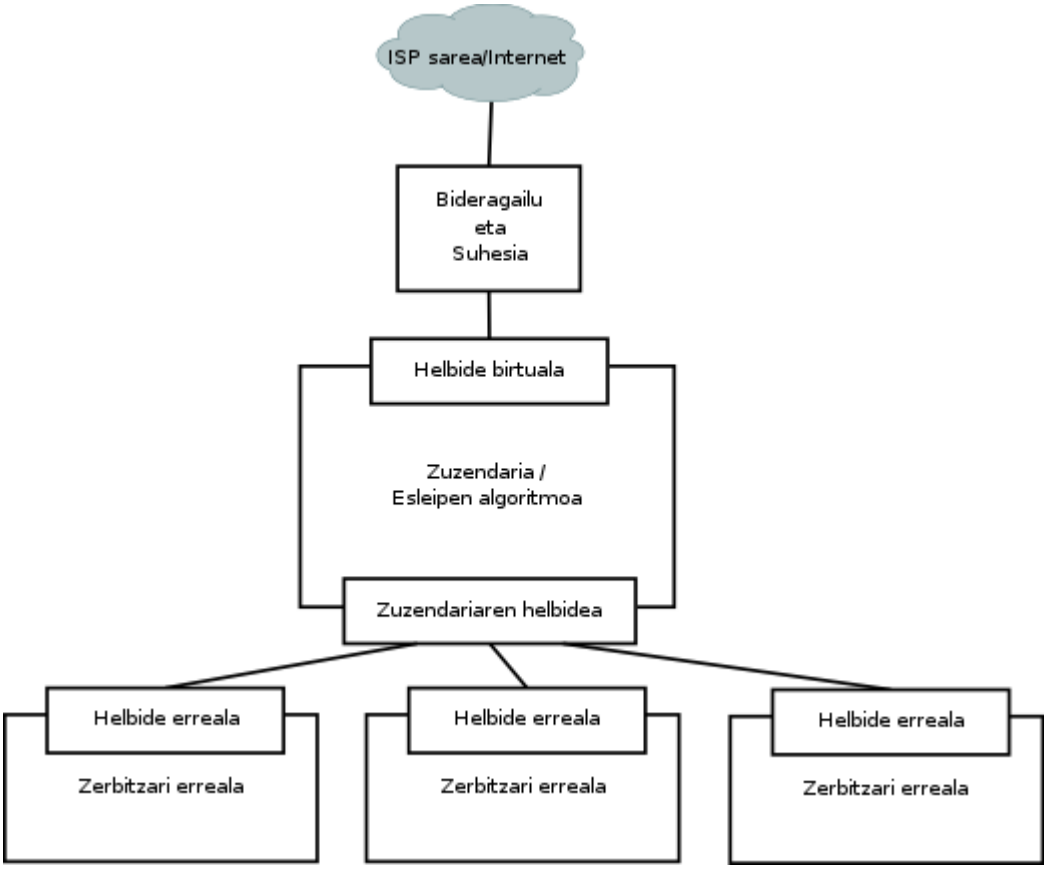
- Karga banatzeko gailua edo zuzendaria[58]: Makina hau izango da zerbitzu eskaerak jasoko dituena. Bere ardura bakarra izango da hauek jaso eta zerbitzari erreal talde baten artean lana banatzea, algoritmo baten arabera.
- Zerbitzari erreala: Zuzendariak jasotzen duen eskaera bakoitza zerbitzari erreal batera bidaltzen du. Azken honek izango du web, FTP, posta edo bestelako eskaera "benetan" zerbitzatzearen ardura.
- Helbide birtuala: Karga banatzeko sistema baten testuinguruan esan genezake helbide birtuala eskaintzen den zerbitzua identifikatzen duen helbidea dela. Zuzendariak izango du honen ardura, eta HSRP/VRRP bidez kontrolatuta egon daiteke nahi izanez gero. Praktikan, Fnneten bezeroek Internetetik web zerbitzaria atzitzen dutenean IP helbide birtual hau erabiliko dute. Zuzendariaren ardura izango da hemendik aurrera zerbitzari erreala behar bezala aukeratzea.
- Zuzendariaren helbidea: Gehienetan zuzendariak bi sare interfaze izaten ditu gutxienez; batean helbide birtuala egon ohi da, eta bestean zuzendariaren helbidea[59]. Azken hau zerbitzari errealen azpisare berdienan dago, eta elkarren arteko komunikazioa ahalbideratuko du.
- Helbide erreala: Zerbitzari errealetan dauden helbideak dira hauek. Zuzendaria arduratuko da helbide birtualera doan eskaera helbide erreal batera bidaltzeaz. Gaur egun, eta erabiliko dugun Linux ingurunean, hau hiru modutan egin daiteke: NAT, bideratze zuzena eta tunelak erabiliz. Aldeko eta kontrako irizpideak "taula 6" taulan ikusi daitezke, eta azalpen grafikoak "irudia 16/17/18" irudietan.
- Esleipen algoritmoa: Zuzendariak esleipen algoritmo bat erabiltzen du eskaera bakoitza zein zerbitzari errealeri bidali behar dion aukeratzeko. Algoritmoaz gain, iraunkortasuna[60] deritzona erabili daiteke helbide konkretu batetik datozen eskaerak beti zerbitzari bakarrera bidaltzeko. Hau egokia da bezero eta zerbitzari errealaren artean saio bat irekitzen denean, eta datuak mantendu behar direnean,

58 Ingeleraz load balancer. Zenbaitetan director ere irakurri ohi da.

59 Zuzendariaren helbidea eta helbide birtuala sare interfaze berdinean egon daitezke arazorik gabe.

60 Ingeleraz persistence hitza erabili ohi da.

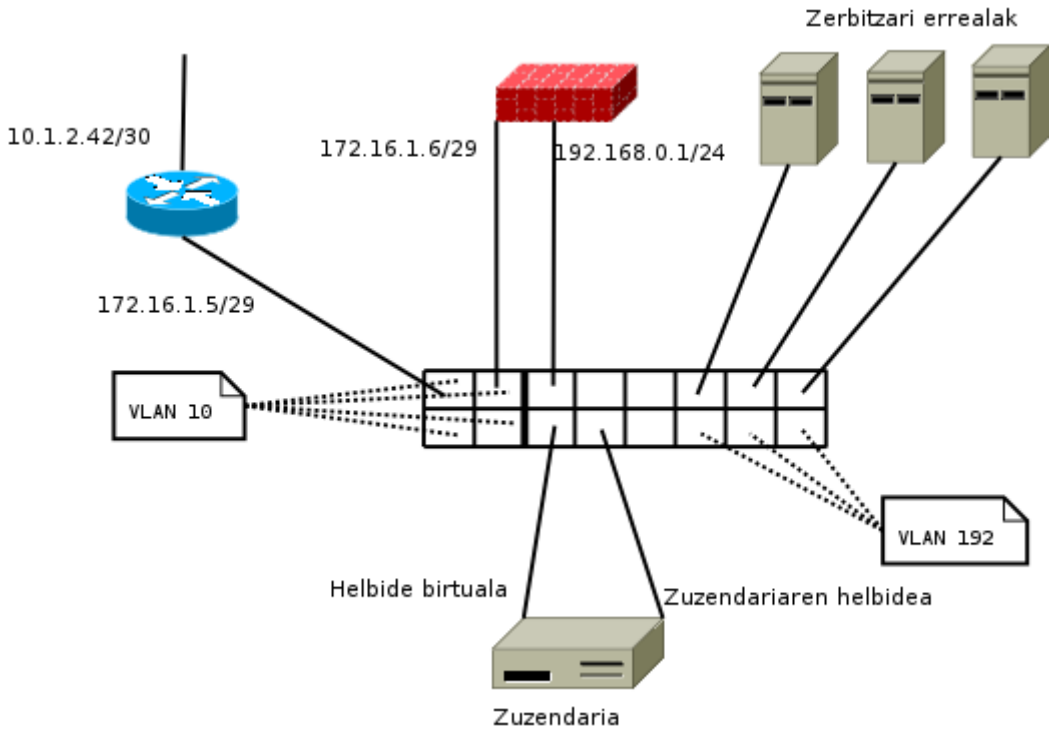
erosketa saskietan gertatzen den bezala. Azken urteotan algoritmo berriak agertzen joan dira; hauetako nagusienen azalpena "taula 7" taulan irakurri daiteke. Gure beharren arabera, kudeatzen duten aplikazioaren arabera, aukeratu beharko dugu zein erabili.



Irudia 14: Karga banatzeko sistemen deskribapena

Bideragailu eta suhesiekin egin genuen bezalaxe, hemen ere kableaketarekin eta erabilitako *switch*-en diseinuarekin zenbait erabaki hartu beharko genituzke. Aukera bat helbide birtuala, zuzendariarena eta errealak *switch* berdinean nahastuta egotea litzateke; edo *switch* berezia izatea zuzendariaren helbide eta zerbitzari errealetarako, edo VLAN desberdinak, ..., azken finean, dagoeneko ezagunak diren beste aukerak.

Erabaki egokia hartzeko zuzendari eta zerbitzari errealen arteko komunikazioa nola egingo dugun jakin beharko genuke. Halere, Fnneten garapen fase honetan *switch* berri eta berezi bat erabiltzea aurrekontuak baztertzen duen zerbait denez, bi aukera baino ez zaizkigu gelditzen; batetik dagoeneko ditugun VLANak erabiltzea, eta bestetik karga banatzeko sistemarentzat berri bat sortzea. Azken hau interesgarria izan daiteke, noski, baina oraingoz azpiegituran behar ez dugun zailtasun maila bat igotzea litzatekeenez, dagoeneko erabilgarri dugun VLAN berdina erabiliko dugu.



Irudia 15: Karga banatzeko sistemaren arkitektura

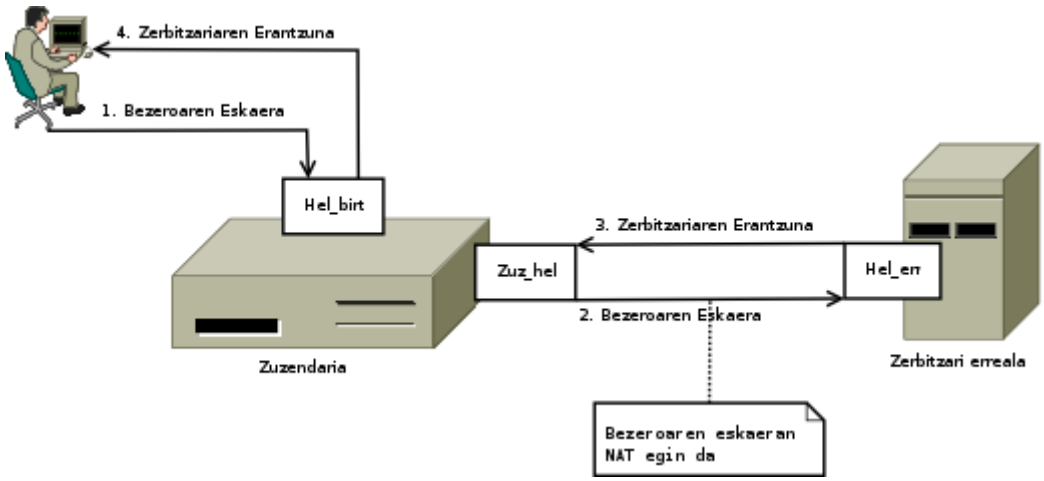
Irudia argiago izateko bideragailu nahiz suhesiaren HSRP/VRRP inplementazioa falta da. Ikusten denez, 192 VLANa erabili dugu zuzendari eta zerbitzarientzat. Zuzendariak bi sare interfaze ditu; hau beti beharrezkoa ez bada ere, malgutasun handia emango digu. Erabiliko ditugun IP helbideak erabakitzeko, aurretik zuzendari eta zerbitzarien arteko

komunikaziorako ditugun aukerak deskribatuko ditut taula eta irudi bidez:

	Ezaugarriak	Aldekoak	Kontrakoak
NAT	Zuzendariak helbide birtuala NAT bidez helbide errealetara itzultzen du.	Zuzendariak IP eta portu batean jasotakoa zerbitzari errealetan beste portu batean egon daiteke. TCP/IP ahalmena duen ia edozer izan daiteke zerbitzari erreala. Inplementatzen erraza.	Zuzendariak bezeroen eskaerak eta zerbitzarien erantzuna jasotzen ditu. Hau eraginkortasun eta fidagarritasun arazoa izan daiteke. Zenbait aplikaziorekin ere arazoak.
Bideratze zuzena	Zuzendari eta zerbitzari errealen artean OSIren 2. maila erabiltzen da.	Bezeroen eskaerak zuzendariak jasotzen ditu, baina zerbitzarien erantzunak ez dira zuzendaritik igarotzen, honek hedagarritasun handia ematen du.	ARP arazoak egon daitezke[61]. Zerbitzari errealetan dagoen sistema eragilea gauza izan behar da arazo honi konponbidea emateko.
Tunelak	Komunikazioa IP tunelak erabiliz egiten da.	Tunelak erabiltzen direnez zerbitzari errealak edonon egon daitezke.	Tunelak sortzeko ahamena duten sistema eragileak baino ezin dira erabili zerbitzari errealetan. ARP arazoak egon daitezke.

Taula 6: Zuzendari eta zerbitzari errealen arteko komunikaziorako aukerak

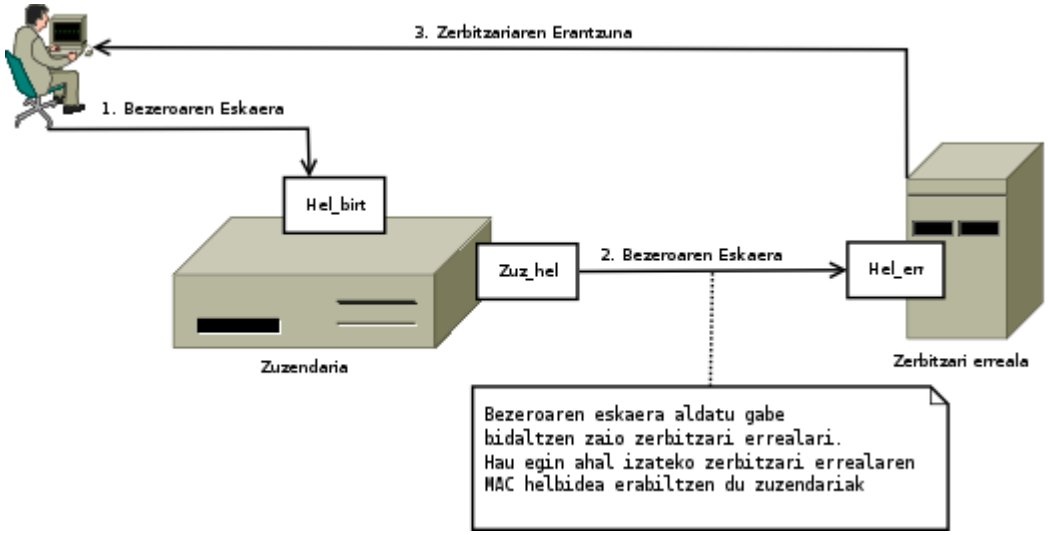
61 ARP arazoen inguruan informazio asko irakurri daiteke http://www.austintek.com/LVS/LVS-HOWTO/HOWTO/ helbidean.



Irudia 16: Komunikazioa NAT erabiliz

NAT prozesu honen inplementazioa egiteko gehienetan azpisare berezi bat erabiltzen da zuzendari eta zerbitzari errealen arteko komunikaziorako. Azpisare hau ia beti pribatua izaten da (RFC 1918), kontutan izanda bezeroak ez duela helbide birtuala (publikoa) baino ikusten. Sistema hau sinplea da, aparteko zailtasunik gabekoa. Arazoen artean aipagarrienetakoa software itxiak batzuetan lizentziak kudeatzeko duen moduarekin dator. Askotan, helbide birtualarekin erlazionatzen da lizentzia bat, baina zerbitzari errealetan ez dago helbide honen arrastorik, beraz, arazoak eman daitezke.

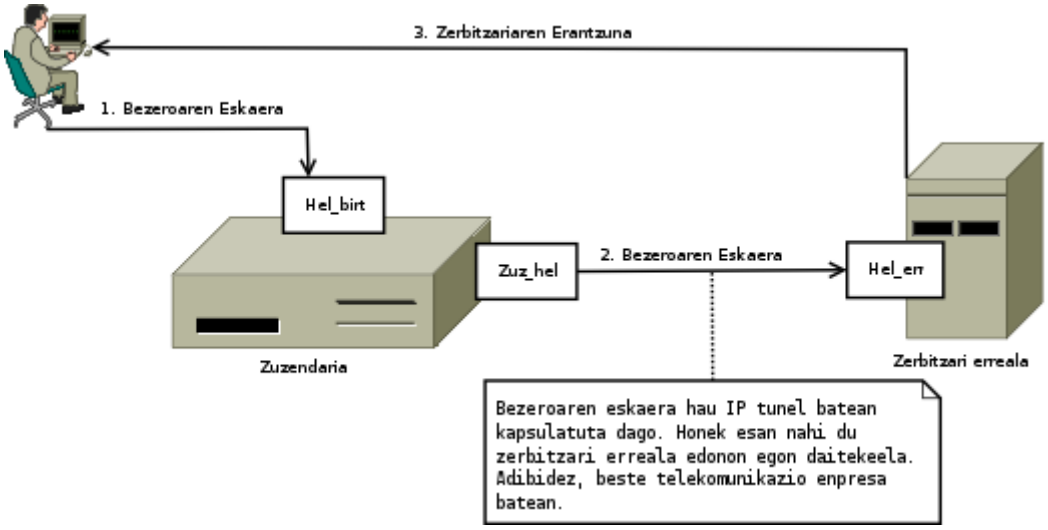
Kernelaren bertsio zaharretan NAT prozesuak eraginkortasun galera bazekarren ere, gaur egun gainditua dagoen arazoa da. Gainera, gaur egungo makinek duten prozesatzeko ahalmenarekin eraginkortasun arazoak, izatekotan, sare txarteletan daude.



Irudia 17: Komunikazioa bideratze zuzena erabiliz

Bideratze zuzenari dagokionean, bezeroak egindako eskaera helbide birtualera joango da, noski, baina erantzuna ez zaio zuzendaritik etorriko, baizik eta zerbitzari errealetik. Honetarako, zuzendariak eskaera jasotzen duenean, aldatu gabe bidaliko dio zerbitzari errealari. Hau egiteko zerbitzari errealaren MAC helbidea erabiliko du. Zerbitzari errealak eskaera hau jasotzen duenean bezeroari erantzungo dio, zuzenean.

Guzti hau egin ahal izateko zerbitzari errealak helbide birtualera doazen eskaerak jasotzeko modua izan beharko du. Praktikan, helbide birtuala interfaze batean izan beharko du (*loopback*, adibidez). Hau honela, helbide birtuala bai zuzendarian bai zerbitzari errealean egongo denez, ARP eskaerekin egon daitezkeen arazoak konpondu beharko dira. Laburbilduz, zerbitzari errealak helbide birtuala izango badu ere, ez du ARP eskaerarik erantzungo, helbide honen ardura zuzendariak baitu.



Irudia 18: Komunikazioa tunelak erabiliz

Tunelek duten abantaiarik nagusiena zerbitzari errealak geografikoki edonon egon daitezkeela da, baina praktikan ez dira gehiegi erabiltzen. Esango nuke gaur egungo joera gehiago dela karga banatzeko sistema lokalak izatea, eta DNS bidez kontrolatzea, bezeroaren arabera, nora bidaliko zaion.

Zuzendari eta zerbitzarien arteko komunikazioa eginda, hurrengo pausoa da bezeroen eskaerak nola esleituko zaizkion zerbitzari bakoitzari. Hau egiteko ondo jakin behar da zein izango den azpian dagoen aplikazioa. Web ingurunetan, esate baterako, bezero eta zerbitzariaren arteko lotura mantendu behar duen erosketa saski baten karga hainbat zerbitzariren artean banatzeko algorimo eta iraunkortasun irizpide batzuk beharko ditugu, eta HTML estatikoa edo irudiak ikusteko baino ez den zerbitzari batean, berriz, beste batzuk. Noizean behin algoritmo berriak agertzen dira, baina praktikan egoera gehienetarako hiru edo lau baino ez dira erabiltzen, nahiz eta gaur egun, 2.6.20 kernelean, guztira hamar dauden. Informazio osoagoa (Lvs, 2006) web gunean eta (Kopper, 2005) liburuan irakur daiteke. Jarraian, taula txiki batean algoritmorik erabilienetakoak:

Algoritmoa	Ezaugarriak
Round Robin (RR)	Zuzendariak eskaera bakoitza zerbitzari errealen artean banatzen du, sekuentzialki. Hau oso egokia izan daiteke zerbitzari guztiak berdinak direnean, eta aplikazio estatikoetarako
Weighted Round Robin (WRR)	Zuzendariak zerbitzari erreal bakoitzari pisu bat ematen dio. Honela, bateko pisua duen zerbitzari batek jasotako eskaera bakoitzeko biko pisua duen batek bi jasoko lituzke. Hau erabilgarria izan daiteke, esate baterako, zerbitzarien hardwarea desberdina denean.
Destination hashing	IP batera doazen eskaerak beti joango dira zerbitzari berdinera. Hau erabilgarria izan daiteke *cache* edo *proxy* sistemekin.
Source hashing	Metodo hau erabiltzen da batez ere zuzendariak ziurtatu behar duenean jasotako eskaera bakoitza etorritako bide berdinetik itzuli behar dela. Hau da, zuzendaria bi suhesi edo bideragailuetara konektatuta dagoenean.
Least Connection (LC)	Eskaera jasotakoan, zuzendariak zerbitzari erreal bakoitzak momentuan dituen konexio irekien arabera egingo du esleipena. Gutxien dituenari bidaliko dio.
Weighted Least Connection (WLC)	Konexio kopuru txikiena eta pisua erabiltzen dira esleipen erabakia egiteko.

Taula 7: Karga banatzeko sistemen esleipen algoritmorik erabilitakoenak

Algoritmoekin batera beste kontzeptu interesgarri bat erabil daiteke; iraunkortasunarena,

hain zuzen ere. Honi esker bezero batek[62] egindako eskaera guztiak denbora tarte batez zerbitzari erreal berdinera joango dira, algoritmoa kontutan hartu gabe. Aipatutako (Kopper, 2005) liburuan zenbait adibide ikus daiteke.

5.6.2.- Oinarrizko inplementazioa

Fnneten liburutegiak bi zerbitzu nagusi izango ditu; batetik web zerbitzari orokorra, enpresari buruzko informazioa, saltzen diren produktuak, bezeroarenganako arreta eta gainontzekoekin, eta bestetik web zerbitzari segurua, erosketak egin ahal izateko. Aurrenekoa TCP 80 portuan egongo da, eta bigarrena TCP 443.ean. Helbide birtualak aukeratzeko orduan, bi zerbitzuak IP bakarrean jartzeko aukera badugu ere, C klase oso bat daukagunez, bi helbide erabiliko ditugu. Honek funtzionaltasunari dagokionean ez du desberdintasunik suposatzen, baina zenbait egoeratan malgutasuna emango digu. Orokorrean, eta helbideak soberan izango ditugula ziurta badezakegu, zerbitzu bakoitza IP batean jarriko dugu. Kontutan izan zuzendari bakar batean helbide birtual guztiak izan ditzakegula arazorik gabe, eta edozein momentutan berriak gehitzeko moduan egongo garela, etenaldirik gabe.

Hurrengo pausoa zuzendari eta zerbitzarien arteko komunikazioa nola egin behar den erabakitzea da. ARP trafikoarekin arazorik ez izateko NAT sistema erabiliko dugu. Garai batean, batez ere oso azpiegitura handietan, ez zen asko erabiltzen eraginkortasun arazoengatik, baina gaur egungo sistema eragile eta makinekin aukera guztiz onargarria da. Edozein kasutan, etorkizunean sistema aldatu nahiko bagenu, aparteko arazorik ez genuke izango, beti ere behar bezalako planifikazioarekin, noski.

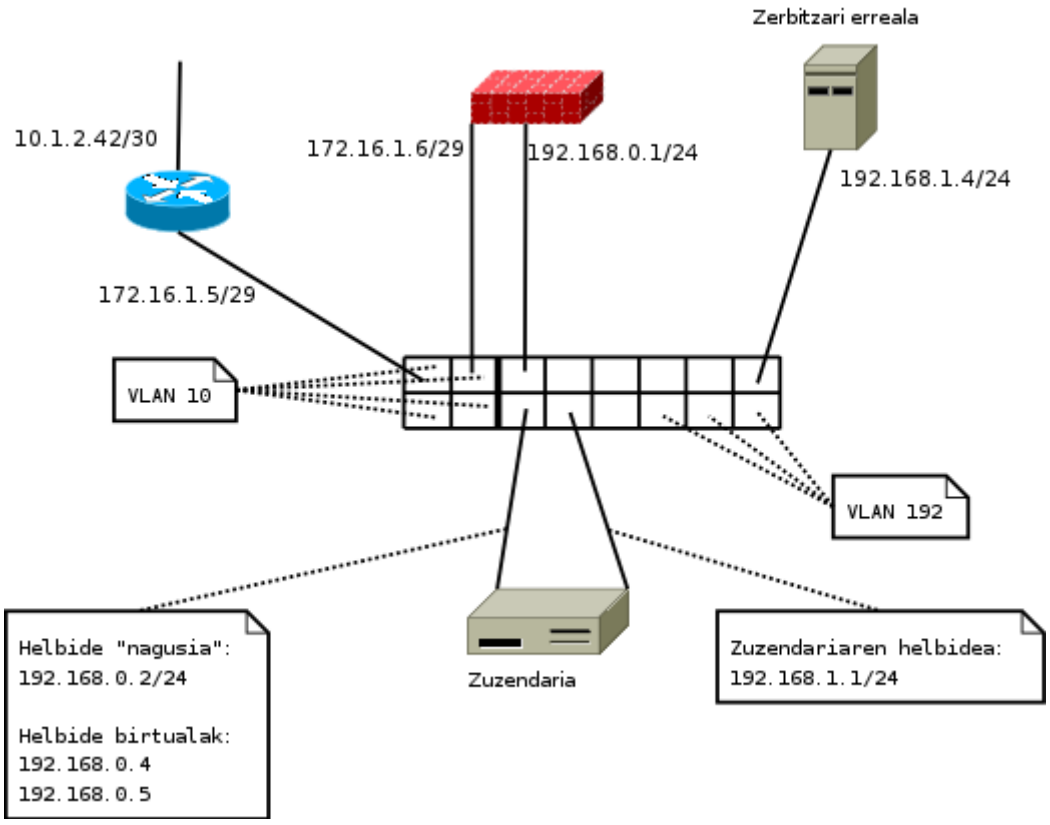
Hasteko, NAT prozesua egin ahal izateko sare bat erabiliko dugu zerbitzari errealetarako. Orain arte erabilitako helbide gehienak ez bezala, sare hau ingurune errealetan ere pribatua izan daiteke. Fnneten kasuan, 192.168.1.0/24 izango da. Gainontzeko datuak

62 Bezeroa izan daiteke IP helbide bat, edo maskara batean sartzen den azpisarea.

ondokoak dira:

- Zuzendaria
  - Helbidea suhesiaren azpisarean: 192.168.0.2.
  - Atebide lehenetsia: 192.168.0.1 (Suhesia).
  - Zuzendariaren helbidea zerbitzari errealen azpisarean: 192.168.1.1.
  - Web zerbitzaria
    - Helbide birtuala: 192.168.0.4.
    - Portua: TCP 80.
  - Web zerbitzari segurua
    - Helbide birtuala: 192.168.0.5.
    - Portua: TCP 443.
- Zerbitzari erreala
  - Helbidea: 192.168.1.4.
  - Atebide lehenetsia: 192.168.1.1 (Zuzendaria).
  - Web zerbitzariaren portua: TCP 80.
  - Web zerbitzari seguruaren portua: TCP 443.

Oraingoz zerbitzari erreal bakarra erabiliko dut; hurrengo ataletan berriak gehituko dizkiot azpiegiturari. Grafikoki ikusita:



Irudia 19: Karga banatzeko sistemaren oinarrizko inplementazioa

Paperean idatzitakoa makinetara eramatea da hurrengo urratsa. Bideragailua sare ahalmena daukan Linux makina da; suhesia sare ahalmena eta iragazpen ahalmena daukan Linux makina da; eta karga banatzeko sistema, noski, hau egiteko ahalmena duen Linux makina ere bada. Guztiak sistema eragile berdinarekin izatea, azpiegitura ahal den homogeneoena izatea, abantaila handia izan daiteke, formaziorako denbora txikiagoa edo mantentze sinpleagoa eskatzen dituelako, besteak beste. Hori bai, beti izan behar da kontutan aurrenekoa gauzak ondo egitea dela, eta ez guztirako Linux erabiltzea. Zorionez, Linux sistema eragilearekin kalitateko bideragailu, suhesi eta karga banatzeko sistemak eraiki daitezke.

Hau honela, *Linux Virtual Server* (LVS) izeneko proiektua erabiliko dugu orain arte

ikusitakoa inplementatzeko. Proiektuak bi osagai nagusi ditu. Aurrenekoa kernelaren parte da 2.4.23 bertsioaz geroztik; bigarrena, kernelarekin komunitzen den aplikazioa da, eta zerbitzari errealak, algoritmoak, iraunkortasuna, eta honezkero aipatutakoak konfiguratzeko erabiltzen da. Kernela konfiguratzeko zenbait banaketetan moduluak aurrez konpilatuta eskaintzen dira. Guk eskuz egin nahiko bagenu "*Networking -> Networking options -> IP: Virtual Server Configuration*" menuaren azpian algoritmo eta beste zenbait aukera izango ditugu eskuragarri. Gomendagarria izaten da algoritmo guztiak modulu bezala konpilatu, eta gero erabiliko direnak kargatzea.

Kernelarekin komunikatzeko aplikazioa *ipvsadm* deitzen da. (Lvs, 2006) Web gunetik jaitsi daiteke *rpm* edo *tar.gz* formatuan. Linux banaketa gehienetan ere izen berdineko paketea instalatu daiteke.

"Irudia 19" irudian dagoena zuzendarian inplementatzeko pauso hauek eman beharko genituzke:

```
# Erabiliko ditugun LVS moduluak kargatzeko, sistemak automatikoki
# egiten ez badu. Bi algoritmo erabiliko ditugu, rr eta wrr:
modprobe ip_vs ; modprobe ip_vs_rr; modprobe ip_vs_wrr

# Sare txartelen konfigurazioa:
ip addr add 192.168.0.2/24 dev eth0
ip addr add 192.168.1.1/24 dev eth1
ip route add default via 192.168.0.1 dev eth0
# HTTP eta HTTPS zerbitzuetarako IP helbideak:
ip addr add 192.168.0.4/24 dev eth0
ip addr add 192.168.0.5/24 dev eth0
echo 1 > /proc/sys/net/ipv4/ip_forward

# LVSren konfigurazioa. Helbide birtualak, portuak eta algoritmoak:
ipvsadm -A -t 192.168.0.4:80 -s rr
ipvsadm -A -t 192.168.0.5:443 -s wrr -p 300

# LVS. Zerbitzari errealak, komunikazio mota eta pisuak:
ipvsadm -a -t 192.168.0.4:80 -r 192.168.1.4:80 -m
ipvsadm -a -t 192.168.0.5:443 -r 192.168.1.4:443 -m -w 2
```

```
# Zerbitzari erreal berriak gehitzeko adibideak:
# ipvsadm -a -t 192.168.0.4:80 -r 192.168.1.<x>:80 -m
# ipvsadm -a -t 192.168.0.5:443 -r 192.168.1.<y>:443 -m -w <z>

# Egindakoa ikusteko:
root@zuzendaria:/# ipvsadm -ln
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  192.168.0.4:80 rr
  -> 192.168.1.4:80               Masq    1      0          0
TCP  192.168.0.5:443 wrr persistent 300
  -> 192.168.1.4:443              Masq    2      0          0
```

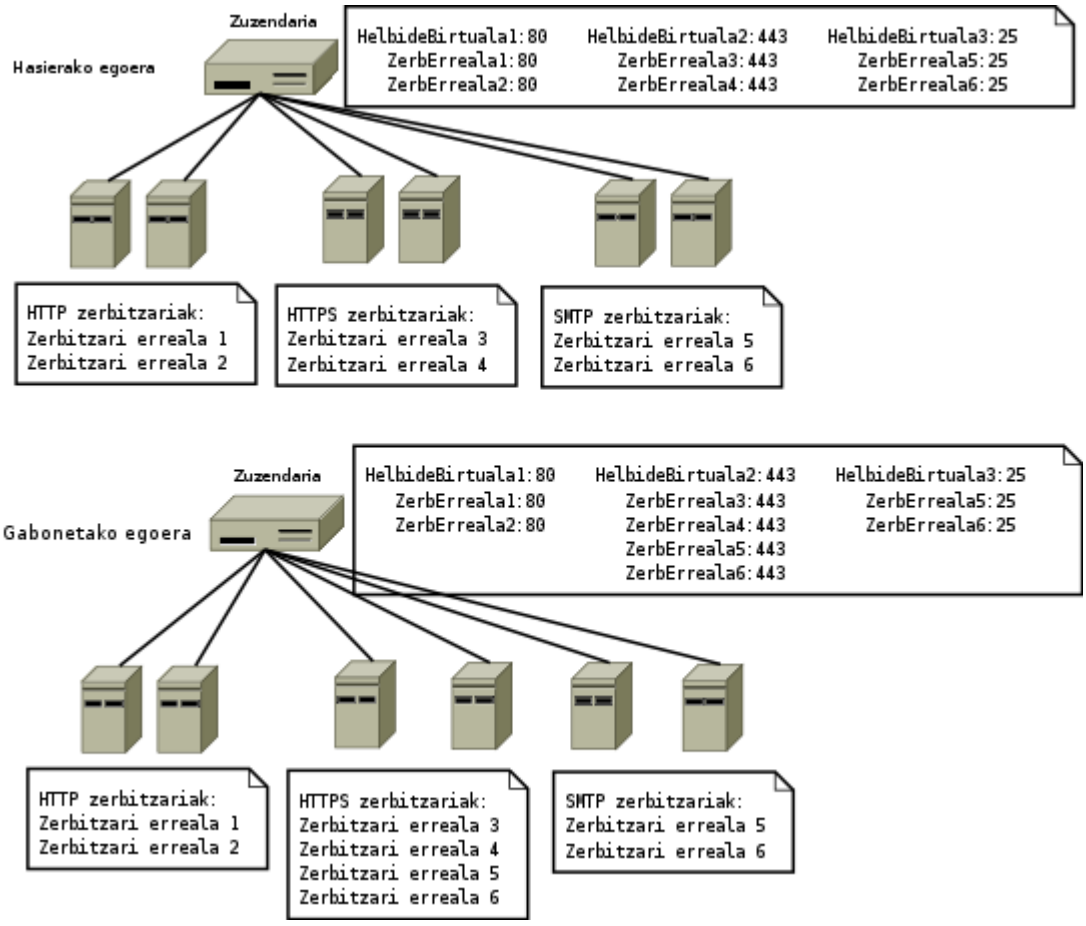
Adibidean bi algoritmo erabili ditut, *round robin* web zerbitzurako, eta *weighted round robin* web zerbitzari segururako. Azken honetan 300 segundutako iraunkortasuna jarri dut, beraz, bezero batek denbora tarte horretan egiten dituen eskaera guztiak zerbitzari erreal berdinera joango dira. Zerbitzari errealak NAT bidez komunikatzen dira zuzendariarekin (*-m*) eta zerbitzari segurura doanak 2ko pisua du (*-w 2*).

NAT sistema erabiltzen denean, zerbitzari errealetan ez da inolako konfigurazio berezirik egin behar. IP helbide egokiak jarrita, eta kontutan izanda atebide lehenetsia zuzendaria dela (192.168.1.1), sistema berehala izango dugu martxan. Modu berean, zerbitzari berriak gehitu edo kentzeko pausoak minutu gutxitako lana da: Batetik, zerbitzari berriari NAT sarearen helbide bat eman, 192.168.1.<x>, atebide lehenetsia zuzendaria dela esan, eta azken honetan erregela berri bat gehituko genuke.

Gehitu bezain erraza da makinak sistematik kentzea. Modu bat zerbitzari erreala konfiguraziotik zuzenean ezabatzea da (adibidez hardware arazo bat egon denean); beste bat 0 pisua ematea da; honela zerbitzari errealak oraindik burutu gabeko eskaerak bukatuko lituzke, baina berririk jaso gabe (planifikatutako geldialdietan).

```
# Zerbitzari erreala kentzeko bi moduak:
ipvsadm -d -t 192.168.0.5:443 -r 192.168.1.4:443
# edo
ipvsadm -e -t 192.168.0.5:443 -r 192.168.1.4:443 -m -w 0
```

Sistema honen abantailak argiak dira. Fidagarritasunaren ikuspuntutik, zerbitzuak hainbat makinatan banatuta izateak zerbitzari errealetan eman daitezkeen arazoen aurrean babesa ematen du. Eraginkortasunaren ikuspuntutik, beharren arabera, zerbitzariak gehitu edo kentzeko aukera izango genuke, inongo etenaldirik gabe. Baliabideen esleipenari dagokionean, karga banatzeko sistemen dinamismoak aukera ematen du momentu batean zerbitzari errealak zerbitzu birtualen artean banatzeko. Adibidez, demagun Fnnetek hiru zerbitzu eskaintzen dituela: HTTP, HTTPS eta SMTP, eta gabonetan HTTPS zerbitzua indartu nahi dugula, lan karga gutxi duten SMTP zerbitzarietaz baliatuta.



Irudia 20: Karga banatzeko sistemen malgutasuna

Honelako egoerak posible izateko zerbitzari guztiek konfigurazio berdina izan behar dute. Irudian agertzen den kasuan, esate baterako, SMTP zerbitzaria duten makinek HTTPS zerbitzaria ere konfiguratuta izan behar dute, nahiz eta normalean trafikorik jasoko ez duten.

Honelako sistemen beste alde positibo bat segurtasunaren aldetik dator, batez ere NAT erabiltzen denean. Honetan, zerbitzari errealek ez dute zuzendarian konfiguratu ez den trafikorik ikusiko.

Aldekoak erraz ikusten diren bezala, kontrakoak ere oso argiak dira. Zer egin zuzendaria matxuratzen bada? Galdera honi erantzuna emateko erabiliko dugun softwarea dagoeneko ezagutzen dugun *Keepalived* da. Izatez, *Keepalived* LVSrentzat propio egindako aplikazioa da, nahiz eta orain arte berezita erabili dugun. Hurrengo atalean sakonago deskribatuko dut *Keepalived* eta LVSren arteko integrazioa.

Bigarren arazoa berria da. Hainbat zerbitzari errealen artean banatuko bada lana, edukia ere berdina izan beharko litzateke. Nola sinkronizatu edukia, batez ere oso aldakorra denean? Guzti hauek ere aurrerago landuko ditugu.

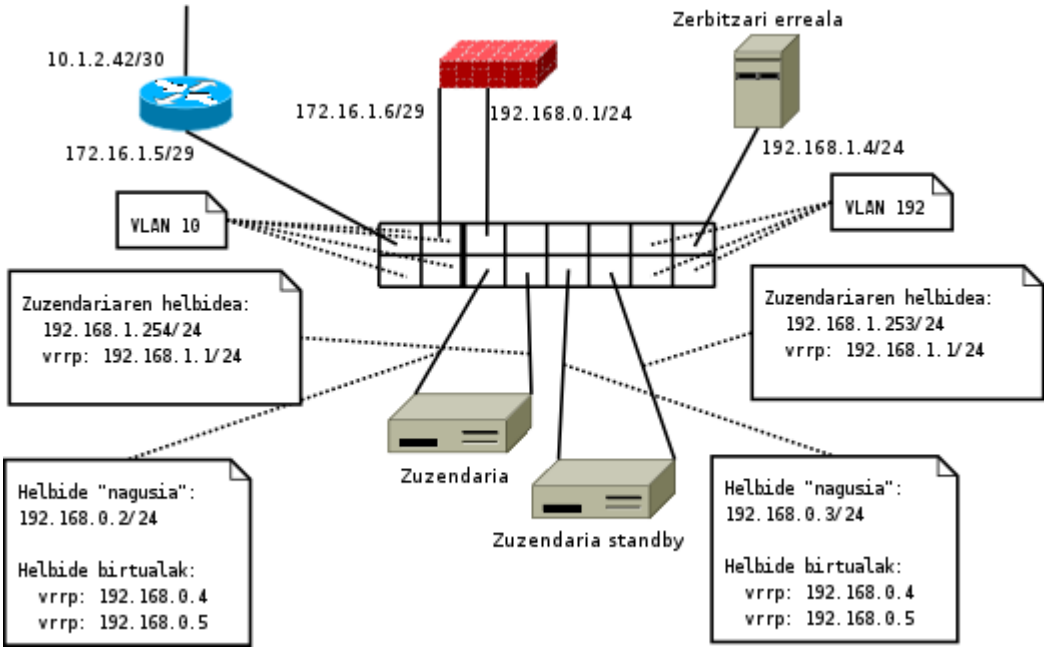
5.7.- Karga banatzeko sistemen garapena

Arkitekturan elementu berri bat gehitzen denean konplexutasun maila igotzeaz gain fidagarritasunean eragin kaltegarria izan dezakeen punturen bat gehitzeko arriskua izaten da. Karga banatzeko sistema gehitu denean egoera honetan aurkitzen gara. Honen aurrean neurririk hartu ezean, zuzendaria hondatuko balitz arazo larrietan aurkituko ginateke, batez ere NAT sistema batean.

Azpiegitura diseinatu denean, hobekuntzak egin direnean, ahal izan den neurrian, homogeneotasuna bilatu da bai hardware bai softwarearen aldetik. Karga banatzeko sistemak ez dakar aparteko desberdintasunik, beraz, lan handiegirik gabe ezarri ahal

izango dugu bigarren zuzendari bat *Keepalived* eta VRRP ezaguna erabiliz. Hau gutxi balitz, *Keepalived* sortu zenean LVS sistemen osagarri bezala pentsatu zenez, oso ondo integratzen da azken honekin.

Keepalived softwarea zerbitzari errealak monitorizatzeko gauza da, eta konfigurazioaren arabera ekintzak burutu ahal izango ditu, hala nola pisuak aldatu edo zerbitzari errealak kentzeko aukera. *Standby* zuzendaria ere sinkronizatuta mantendu dezake *takeover*[63] egoeretan konexiorik gal ez dadin. Lortu nahi dena grafikoki



Irudia 21: Karga banatzeko sistema eta VRRP

Irudia argiagoa izateko bideragailu eta suhesi bakarra jarri dut. Zuzendarian zerbitzuetarako diren IP helbide birtual guztiak VRRP bidez kontrola daitezke, zuzendariaren helbidearekin gertatzen den bezalaxe. Hau egiteko ohiko sintaxia duen konfigurazio fitxategia idatzi behar da. Honekin batera zerbitzari errealetan dauden

63 *Takeover* hitza erabiltzen da *backup*-zuzendariak funtzionamenduan dagoenaren tokia hartzen duenean.

zerbitzuak monitorizatzeko atalak gehitu dizkiogu azken fitxategi honi.

```
global_defs {
   notification_email {
     abisuak@posta_domeinua
   }
   notification_email_from abisuak@posta_domeinua
   smtp_server posta_zerbitzaria_ip
   smtp_connect_timeout 30
   router_id LVS_DEVEL
}

vrrp_sync_group VG1 {
    group {
     VI_1
     VI_2
    }
}

vrrp_instance VI_1 {
    state MASTER
    interface eth0
    smtp_alert
    virtual_router_id 51
    priority 150
    advert_int 1
    virtual_ipaddress {
        192.168.0.4
        192.168.0.5
    }
}

vrrp_instance VI_2 {
    state MASTER
    interface eth1
    smtp_alert
    virtual_router_id 52
    priority 150
    advert_int 1
    virtual_ipaddress {
        192.168.1.1
    }
```

```
}

virtual_server 192.168.0.4 80 {
      delay_loop 30
      lb_algo rr
      lb_kind NAT
      protocol TCP
      sorry_server arazoak_daudela_abisatzeko_zerbitzaria 80

      real_server 192.168.1.4 80 {
            weight 1
            HTTP_GET {
                  url {
                        path /probak.html
                        digest ab84a42b29cd82f5ecaf213ac9ebb02
                  }
            connect_timeout 3
            nb_get_retry 3
            delay_before_retry 2
            }
      }
}

virtual_server 192.168.0.5 443 {
      delay_loop 30
      lb_algo wrr
      lb_kind NAT
      persistence_timeout 300
      protocol TCP
      sorry_server arazoak_daudela_abisatzeko_zerbitzaria 80

      real_server 192.168.1.4 443 {
            weight 2
            TCP_CHECK {
                  connect_timeout 3
            }
      }
}
```

Konfigurazio fitxategi hau zuzendari nagusiari dagokio (*state MASTER*). Besteari dagokion konfigurazioan aldatu beharreko bakarrak egoera (*state*) eta lehentasun balioa (*priority*) dira, suhesiekin egin genuen bezalaxe. Zerbitzari birtualei dagozkien atalak

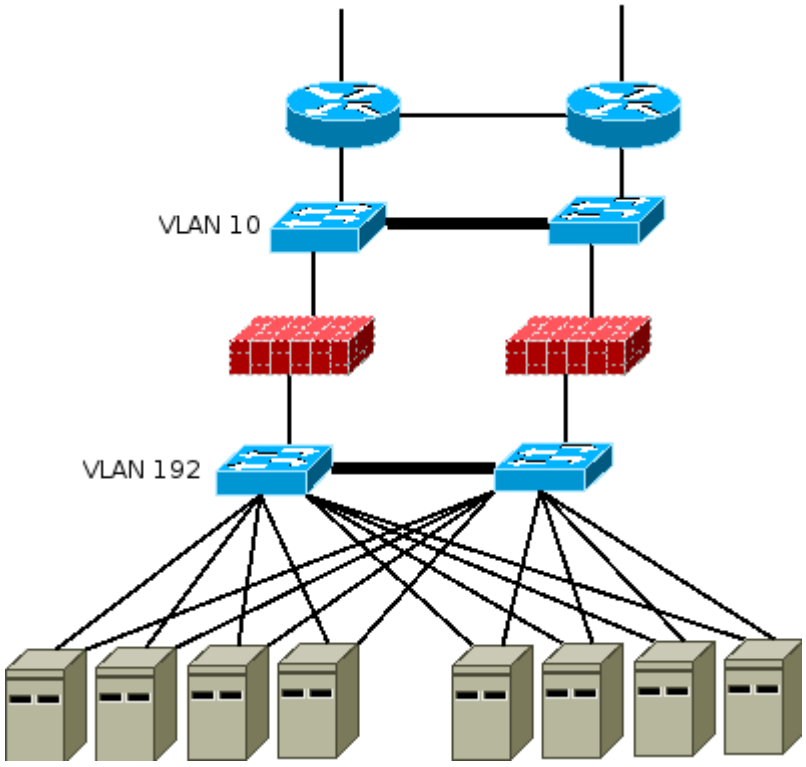
berdinak dira bi zuzendarietan. Aipagarria da *sorry_server* aukerarekin zerbitzari erreal guztiak hondatu direnean errore mezu bat emateko zerbitzaria konfiguratu daitekeela ("momentu honetan mantentze lanak egiten gabiltza" eta honelako mezuak). HTTP_GET eta TCP_CHECK zerbitzari errealak monitorizatzeko *script*-ak dira. Lehenengoak web eskaera bat egiten du, eta jasotako erantzunaren MD5 balioa konfiguratutakoaren berdina bada zerbitzaria ondo dagoela suposatuko du. Bigarrenak TCP konexio ahalegin arrunta egiten du.

5.8.- Bideratze, suhesi eta karga banatzeko sistemak. Azken ideiak

Orain arte egindako hobekuntzak fase desberdinetan sailkatu dira. Hauetako bakoitzak helburu bat zuen, azpiegituraren atal zehatz batekin zegoen erlazionatuta. Ingurune erreal batean, hau egiten hasi aurretik helburu orokorra ere finkatuta egotea gomendagarria izaten da, baina proiektu honetan erabilitako teknologien azalpena egiteko beharrak derrigorrez atzeratu du arkitekturaren azken helburu honen deskribapena.

Oraindik ere sareko hardwarerik ez dugu aztertu, eta ez dugu aztertuko proiektu honen esparrutik kanpo gelditzen delako. Gaur egun hardware fabrikatzaile askok hainbat aukera ematen dute eraginkortasun, fidagarritasun edo segurtasuna bezalakoak hobetzeko; batzuk estandarretan oinarrituta, beste batzuk ez, HSRP eta VRRPrekin ikusi dugun bezala. Hemen diseinu mailako ideia orokor batzuetara mugatuko naiz. Informazio gehiagorako fabrikatzaileen web guneak hasiera ona izan daiteke, eta zertifikaziorako eskaintza duten fabrikatzaileen kasuan, honetarako prestatutako liburuak ere gomendagarriak izaten dira. *Cisco*-ren kasuan, adibidez, *Cisco Press* (Ciscopress, 2007) argitaletxeak hamaika liburu ditu argitaratuta, eta CCDA eta CCDP zertifikazioak ditu eskuragarri sareen diseinuari dagokionean. Beste argitaletxe askok ere liburuak argitaratuta dituzte gai hauen inguruan.

Sareko azpiegituran, hardwarean, fidagarritasuna bilatzen denean sarearen diseinu hierarkikoaz baliatuko gara. Honela, maila bakoitzean fidagarritasun neurri desberdina bila dezakegu, beti ere aurrekontuaren arabera. Gehienetan gailuen artean konexio fisiko asko izan nahiko genituzke, baina aurrekontuak seguraski nahi baino gutxiago egiteko aukera emango digu, eta gainera konfigurazioa ahal dugun sinpleena mantendu beharko genuke. Esango nuke normalean tarteko konponbideak bilatu behar izaten direla, ahalik eta fidagarritasun maila altuena eta konfigurazio eta aurrekontu txikienarekin. Fnneten kasuan, esate baterako, jarraian datorren irudiak oinarrizko fidagarritasuna ematen digu konfigurazioa simple mantenduta.



Irudia 22: Sarearen fidagarritasuna osatuta

Hasteko, orain arte *switch* berdinetan egindako VLANak banatu ditut irudiari argitasuna emateko, baina berdin berdin egin zitekeen *switch* bikote bakarrarekin. Ikusten denez,

diseinu honetan gailu BAKAR baten matxurak ez luke ondorio larririk ekarriko, eta konfigurazioari dagokionean aparteko zailtasunik ez luke gehituko. Lerro lodiz marraztutako *switch*-en arteko loturetan *Etherchannel*[64] eta VTP[65] erabiliko genituzke, baina hortik aurrera dagoeneko landu dugun teknologia baino ez litzateke erabiliko. Fnneten ardura izango da eskatutako fidagarritasun maila zehaztea. Irudian egindakoa aukera bat baino ez da; planteatutakoa *Cold-standby* eta N+1 gailuak erabiliz oraindik ere sinpleagoa izan daitekeen bezalaxe askoz konplexuagoa ere izan zitekeen.

64 Etherchannel Cisco Systems enpresaren protokoloa da, IEEEren estandarra 802.3ad.

65 Irudian ez da derrigorrezkoa, baina honelako loturetan ohikoa izaten denez aipatu dut. Hemen ere Cisco-k bere ISL dauka, eta IEEEk 802.1q estandarra.

6.- ZERBITZU NAGUSIEN GARAPENA

Atal honetan sare azpiegitura gehienetan aurki daitezkeen zerbitzu nagusien hobekuntza prozesua egingo dut. Datu-base kudeaketa-sistemekin hasiko naiz, eta gero fitxategien biltegiratzea egiteko makinekin jarraituko dut.

6.1.- Datu-base kudeaketa-sistemak

Gaur egun, esan daiteke ia edozein ingurunetan datu-baseak erabiltzen direla. Proiektu honetan landutako web ingurunean ere, hauen beharra nabaria da. Liburuen katalogoa, bezeroen datuak edo egindako eskaerak, besteak beste, datu-base kudeaketa-sistema batean egongo dira. Ikusten denez, biltegiratutako datu hauen garrantzia izugarria da.

Softwarearen aukeraketa egiteko orduan eskaintza zabala da, bai software librean bai software itxian. Egokiena zein den esatea ez da batere erraza. Aholkularitza lana behar bezala egiteko, tresnak, merkatua, ezagutu behar da, eta gero aukeratu zein komeni zaigun gehien, baina proiektu honetan software libreaz ari naizenez bi baino ez ditut aipatuko; *Mysql* eta *Postgresql*. Bi hauek dira, seguraski, maila honetan historikoki gehien erabili direnak, baina ez bakarrak. Azken urteotan, esate baterako, *Mysql*-ek *MaxDB* deritzona mantentzen du SAP[66] ingurunetan lan egiteko.

Mysql izango da hemendik aurrera landuko dudan datu-base kudeaketa-sistema (DBKS). Gaur egun, software libreaz hitz egiten denean berehala agertzen da *Mysql* punta puntako erreferentzia bezala. Izatez, gaur egun bertsio librearekin batera beste lizentzia komertzial bat ere badu enpresei eguneraketa automatikoa edo aholkularitza teknikoaren moduko zerbitzuak emateko. Garapenaren aldetik, garai batean azkartasuna lortzeko abantaila nagusiena zuen sinpletasuna ezin da luzaroago erabili. Bertsio berriekin batera SQL galdera bakarrean kateaketak egiteko aukera, trantsakzioak, prozedurak eta fidagarritasuna hobetzeko aukerak gehitu dira, besteak beste. Azkeneko hauek dira, hain

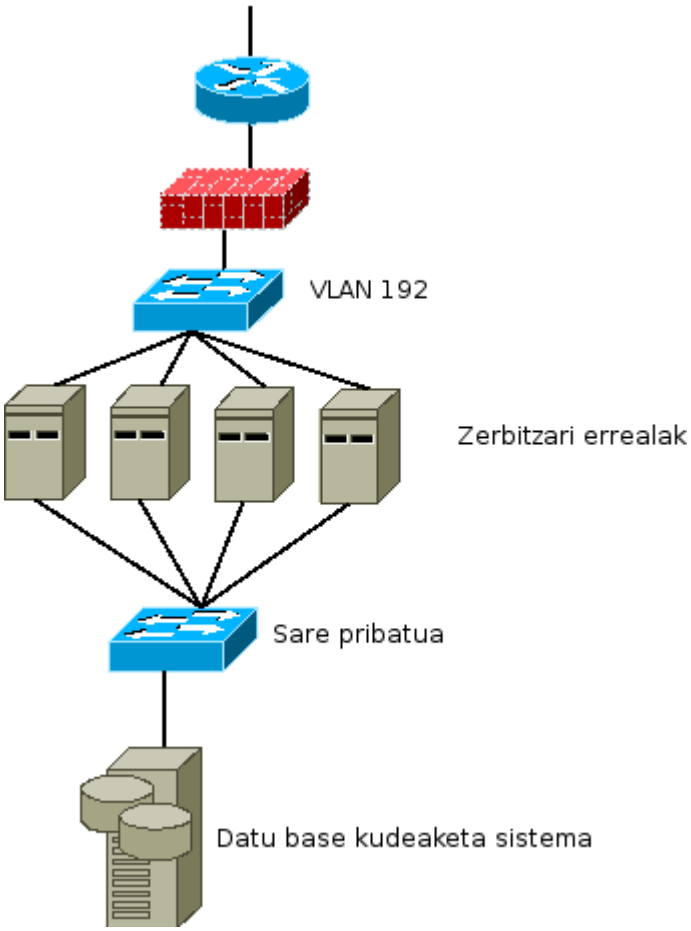
66 http://www.sap.com

zuzen ere, proiektu honetan interesatzen zaizkigunak. *Mysql* erabiliko dugu datu-base kudeaketa-sistemen fidagarritasuna eta eraginkortasuna handitzeko gehienetan erabiltzen diren tekniken deskribapena egiteko. Fnnetek *Mysql* ordez beste bat erabili izan balu inplementazioa desberdina izango litzateke, baina kontzeptu berdintsuak erabiliko genituzke.

6.1.1.- Diseinu arau orokorrak

Orokorrean, eta are gehiago web ingurunetan, datu-baseen atzipena pribatua izaten da. Fnneten kasuan, esate baterako, web zerbitzariek izango dute datu-basean dagoen informazioa argitaratzearen ardura. Fnneti dagokionean, ez da zerbitzari propioetatik ez datorren beste inolako atzipenik egongo datu-base kudeaketa-sistemara, eta hau kontutan izan beharko genuke azpiegitura osatzeko orduan. Honen islada naturalena da DBKS izango duen makina bideratze pribatua duen sare batean jartzea. Honekin batera, suhesi nagusian, makinan eta baita aplikazioan ere segurtasun neurriak hartu beharko genituzke.

Sare pribatu hau bezeroekin harremana izateko den sarearengandik isolatzeko moduak dagoeneko erabili ditugu; batetik VLAN berri bat sortzeko aukera izango genuke, eta bestetik *switch* berri bat erabili genezake. Edonola ere, zerbitzari errealetan bigarren sare txartel bat jartzea derrigorrezkoa bezain gomendagarria egiten da, trafiko pribatua eta publikoa erabat banatzeko. Kostuaren aldetik, sare txartelak merkeak dira gaur egun, eta konfiguratzen errazak.



Irudia 23: DBKS makina azpiegituran

Karga banatzeko sisteman NAT prozesua egiteko IP helbide pribatuak erabiltzen bagenituen ere, zerbitzari errealak VLAN 192an bertan jartzea erabaki zen. Orain, ordea, helbideratze pribatua erabiltzen duen DBKS makina beste *switch* batean jartzea erabaki da (VLAN berezi bat ere izan zitekeen *switch*-aren ordez). Zergatik? Hasteko, zerbitzari erreal eta datu-basearen artean sortutako trafikoa pribatua da, ez bakarrik helbideratzeari dagokionean, baita kontzeptualki ere. Karga banatzeko sistemarako NAT erabili dugunez, zerbitzari errealetarako ere IP pribatuak erabili ditugu, baina beste edozein metodo erabilita hau ez litzateke honela izango. Gainera, kontzeptualki ere bezeroekin

erlazionatutako trafikoa izango da. Honekin batera, erabakia hartu zenean azpiegitura gehiegi ez korapilatzea erabaki genuen. Segurtasunari begira ere trafiko motak banatzeak erraztasunak ematen ditu, neurri bereziak hartzeko aukera hobeagoa ematen duelako, "7.2 atalean" ikusiko den bezala.

6.1.2.- Fidagarritasuna eta eraginkortasuna datu-base kudeaketa-sistemetan

Dagoeneko aipatu dugunez, datu-basetan gordetzen den edukia enpresan dagoenaren garrantzitsuenetakoa da. Aplikazioaren arabera (Fnneten kasuan web aplikazioa) dinamikotasun handia izan dezake, bai bezero berriak sortzen direlako edo eskaeraberriak egiten direlako. Datuak etengabe aldatzeak segurtasun kopia egokiak egitea zailduko du, zalantzarik gabe, baina honek ez du esan nahi egin behar ez direnik, nahiz eta hurrengo puntuetan edukia hainbat makinaren artean banatuko dugun. Beraz, egin beharko litzatekeen aurreneko pausoa segurtasun kopiak planifikatzea da, hauetan biltegiratutakoa berehala zaharkituta geldituko bada ere, ezer ez eukitzea baino hobea izango baita, beti. Kopiak egiteko prozesua DBKSren menpekoa da; batzuetan fitxategiak kopiatzearekin nahikoa izaten bada ere, gehienetan, eta batez ere gaur egungo sistema modernoetan, softwarearekin batera aplikazioak etortzen dira. *Mysql*-ek, esate baterako, beste askoren artean *mysqldump* komando ezaguna dauka kopiak egiteko.

Segurtasun kopien inguruan plangintza ona dagoenean, hobekuntzekin hasteko momentua da. Hasierako planteamendua orain arte landutako kontzeptu berdina da, hots, lan karga hainbat zerbitzariren artean banatzea. Halere, datu-base kudeaketa-sistemek duten egiturak karga banatze arrunta zailago jartzen du, jarraiko zerrendan ikusten denez:

- Irakurketarako bakarrik erabiltzen den datu-basea hainbat makinatan kopiatu daiteke arazorik gabe, eta karga banatzeko sistema bidez zerbitzatu. Esan beharrik ez dago, halere, normalean datu-base gehienetan irakurketak eta idazketak egin behar izaten direla, baina zenbaitetan egoera hau eman daitekenez aipamena merezi du. Berez, arazo nagusia idazketak karga banatzeko zuzendariaren azpiko

makina guztietan aldi berean egin beharko liratekeela da. Teorian hau programazio mailan egingarria bada ere, praktikan alferrikako lan asko ekartzen du.

- Zer gertatzen da datu-baseen fitxategiak zerbitzari komunean jartzen badira, esate baterako NFS erabiliz? Datu-base kudeaketa-sistemek *buffer*-ak, erakusleak, semaforoak eta beste hainbat egitura maneiatzen dituzte; bakoitzak bereak, nahiz eta edukia konpartituta egon. Textuinguru honetan arazo asko eman daitezke, makina batean egindako eta gainontzekoetan ikusten ez diren idazketekin hasita eta zeharo hondatutako datu-baseekin bukatuta. Irakurketekin hau ondo joateko aukera handiagoa badago ere, praktikan, eta software librearen inguruan, hau ez da egiten.
- Nagusi/morroi[67] egitura bat erabiliz, idazketak makina nagusi bakarrean egiten dira. Beste N makina izango ditugu morroiaren papera egiten. Nagusian egindako idazketak DBKSk dituen mekanismoen bidez kopiatuko dira morroietara. Kopiaketa hau gehienetan oso azkarra izango bada ere, sareko hardwarearen edo makinen abiaduraren menpe egongo da. Idazketak makina bakarrean egiten direnez, aplikazioa gauza izan behar da idazketak nagusira bidaltzeko, eta irakurketak morroien artean banatzeko. Azken hauek karga banatzeko sistemaren kontrolpean egon daitezke, baldin eta ziurtatzen bada irakurketa eragiketak baino ez zaizkiela pasatzen. Nagusia, halere, karga banatzeko sistematik kanpo egongo da. Nagusiaren rola egiten duen makina hondatuko balitz, morroi batek hartuko luke idazketak jasotzearen ardura. *Debian* banaketan dagoen *Mysql* paketeak nagusi/morroi arkitektura bat osatzeko aukera eskaintzen du ezer berezirik instalatu gabe. *Postgresql*-rekin ere egin daiteke, (Slony, 2007) moduko tresnekin.
- *Cluster* edo nagusi asko izateko aukera ere badago, baldin eta datu-base kudeaketa-sistemaren kontrolpean gelditzen bada idazketak makinen artean sinkronizatzearen ardura. *Mysql*-ek bere 4.1 bertsioaz geroztik hau egiteko aukera eskaintzen du. 5.1 bertsioak hobekuntza handia dakar *clustering*-aren inguruan,

67 Master/Slave ingeleraz.

biltegiratutakoa fitxategietan izateko aukera ematen duelako[68].

Fnneten datu-baseak dinamikoak dira, aldatzen dira. Segurtasun kopiak egitea interesgarria bada ere, DBKS makina bakarrean izateak arrisku handia dakar. Kasu honetan N+1 makina bat izateak ez du aparteko erabilgarritasunik, segurtasun kopia eta makinaren matxuraren artean egindako eskaerak ezin direlako galdu. Hau honela, fidagarritasun eta eraginkortasunaren ikuspuntutik, argi dago makina bat baino gehiago behar dela. Horren argi ez dagoena da nagusi/morroi edo "nagusi/nagusi" *cluster*-a behar denik. Azpiegitura martxan jartzen denean nagusi eta morroi banarekin nahikoa izango da, baina denborarekin, arrakastarekin batera gerta liteke makina bakar batek idazketa eragiketa guztiak jasotzeko ahalmenik ez izatea.

Aurrekontu beharren aldetik nagusi/morroi sistemak hasieran bi makina baino ez ditu eskatzen. Nagusi/nagusi *cluster* batean, ordea, erredundantzia minimoa emateko lau makinatik gora izatea gomendatzen da, nahiz eta hiru nahikoa izan daitekeen. Eraginkortasunari begira, nagusi/morroi sisteman morroi asko izan daitezke, baina nagusi bakarra. *Cluster* batean muga hau desagertzen da. Zein aukeratu? Seguraski hasieran nagusi/morroi izango da aukerarik egokiena, baina etorkizunean *cluster* batean eraldatzeko pausoak argi izan beharko lirateke.

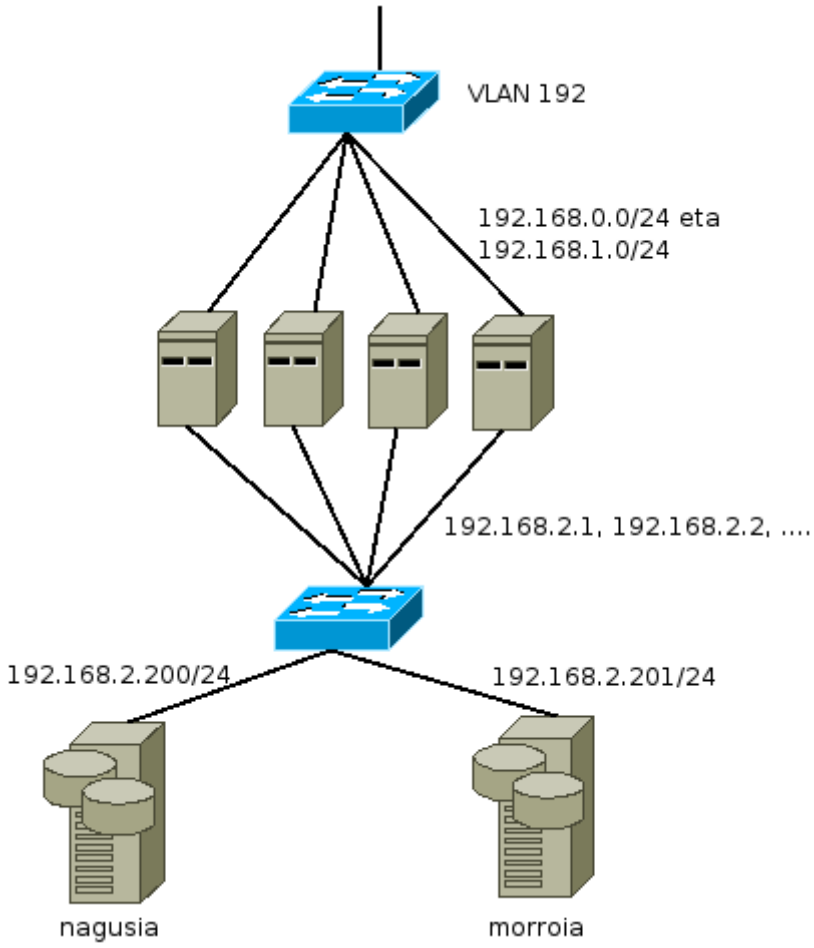
6.1.3.- Oinarrizko inplementazioa

Inplementazioaren aurreneko pausoa dagoeneko landu dugu; sare pribatua, alegia. Hemendik aurrera, kontutan izan behar dugu hartutako erabakiek web aplikazioetan (Fnneten kasuan) eragina izan dezaketela. Jarraian, Fnneten eraginkortasun eta fidagarritasun handiko *Mysql* softwarean oinarritutako datu-base kudeaketa-sistema inplementatzeko oinarrizko ideiak aipatuko ditut, bai nagusi/morroi, bai *cluster*-arekin.

68 4.1 eta 5.0 bertsioetan cluster-ean dauden datu guztiak memorian egon behar dira. 8GBtako datu-base batek, esate baterako, gutxienez 8GBtako memoria duen cluster-a behar du. 5.1 bertsioarekin hau gainditu da.

6.1.3.1.- Nagusi/morroi Mysql erabiliz

Mysql software ezaguna Linux banaketa gehienetan dago eskuragarri. Hemendik aurrerako guztiak 5.0 bertsioa erabiliz egingo ditut. *Debian-testing* erabiliz gero, *mysql-server-5.0* izeneko paketea aukeratuko dugu. Konfiguratzen hasi aurretik, erabiliko ditugun IP helbideak zehaztuko ditut irudi honetan:



Irudia 24: Nagusi/morroi diseinua

Morroi makinekin ez dut karga banatzeko sistemarik erabiliko; ez ezin delako erabili,

baizik eta atal honetan landu nahi dudan kontzeptua datu-baseak direlako. Bezeroek datu-basearekin konexio bat ireki nahi dutenean, idazketetarako 192.168.2.200 zerbitzaria erabiliko dute, eta irakurketak egiteko 192.168.2.200 bera edo 192.168.2.201.

Nagusiaren arazoen aurrean, ba al dugu VRRP protokoloa erabiltzerik nagusiaren IP helbidea (eta honekin idazketak) morroiari esleitzeko? Erantzuna baiezkoa da, baina kontu handiz. 192.168.2.200 helbidea VRRPren kontrolpean egon liteke, eta honen arazoen aurrean morroi baten eskura pasatzeko konfigura genezake, baina jatorrizko nagusia konpondutakoan ezingo luke berriz helbidearen kontrola jaso hondatuta egon den denbora tartean egindako idazketak sinkronizatu arte. Hau VRRPrekin edo *Heartbeat*-ekin (Linux-ha, 2005) egin daitekeen gauza da, baina proiektu honetan ez dut erabiliko.

Mysql-rekin nagusi/morroi sistema bat konfiguratzea oso gauza erraza da. *Debian* ingurunetan *Mysql*-ren konfigurazio fitxategi nagusia */etc/mysql/my.cnf* izaten da. Nagusiaren konfigurazioari dagokionean lerro hauek interesatzen zaizkigu:

```
server-id               = 200
log_bin                 = /var/log/mysql/mysql-bin.log
expire_logs_days        = 7
max_binlog_size         = 200M
#binlog_do_db           = include_database_name
#binlog_ignore_db       = include_database_name
```

Noski, *Mysql*-ren konfigurazioa askoz luzeagoa da, baina nagusian erreplikazioa egin ahal izateko oso gutxi behar dugu. Batetik log bitarra[69], nagusian egindako idazketa eragiketak gordetzeko, eta bestetik zerbitzariaren identifikadorea, bakarra eta beste makinengandik desberdina. 200 jarri dut makinaren IP helbidearekin erlazionatzeko, baina edozein jarri nezakeen. Ez bada ezer jartzen, modu lehenetsian, identifikadorea 1 izango da. Gainontzeko lerroak egin daitezkeen gauzen adibideak baino ez dira.

69 Binary log

Hau eginda, morroiek log bitarraren datuak jaso ahal izateko *Mysql* erabiltzaile bat sortu beharko dugu nagusian.

```
mysql> GRANT REPLICATION SLAVE ON *.*
    -> TO 'morroikopia'@'192.168.2.201' IDENTIFIED BY 'pasahitza';
```

Agindu honek *replication slave* deritzon baimena ematen dio morroi makinari "morroikopia" erabiltzaile eta "pasahitza" gakoarekin. Esan beharrik ez dago erabili dudan gakoa segurtasunaren aldetik ez dela batere egokia.

Nagusiari dagokionean ez da besterik behar. Morroiaren aldean */etc/mysql/my.cnf* fitxategian hauek agertu beharko lirateke.

```
server-id               = 201
log_bin                 = /var/log/mysql/mysql-bin.log
expire_logs_days        = 7
max_binlog_size         = 200M
#binlog_do_db           = include_database_name
#binlog_ignore_db       = include_database_name

master-host = 192.168.2.200
master-user = morroikopia
master-password = pasahitza
```

Azkeneko hiru lerroak dira hemen interesgarrienak; nagusia non dagoen, zein den erabiltzailea, eta pasahitza. Zerbitzariaren identifikadorea hemen ere IP helbidearekin erlazionatu dut. Nagusi eta morroi guztien identifikadoreak desberdinak izan behar dira. Log bitarra morroietan ere izatea gomendagarria da, batez ere nagusiaren rola hartzeko prest izan nahi badugu. Honetan gehiago sakontzeko dokumentazio asko dago. Hasiera puntu bat (Mysql, 2006) web gunea izan daiteke.

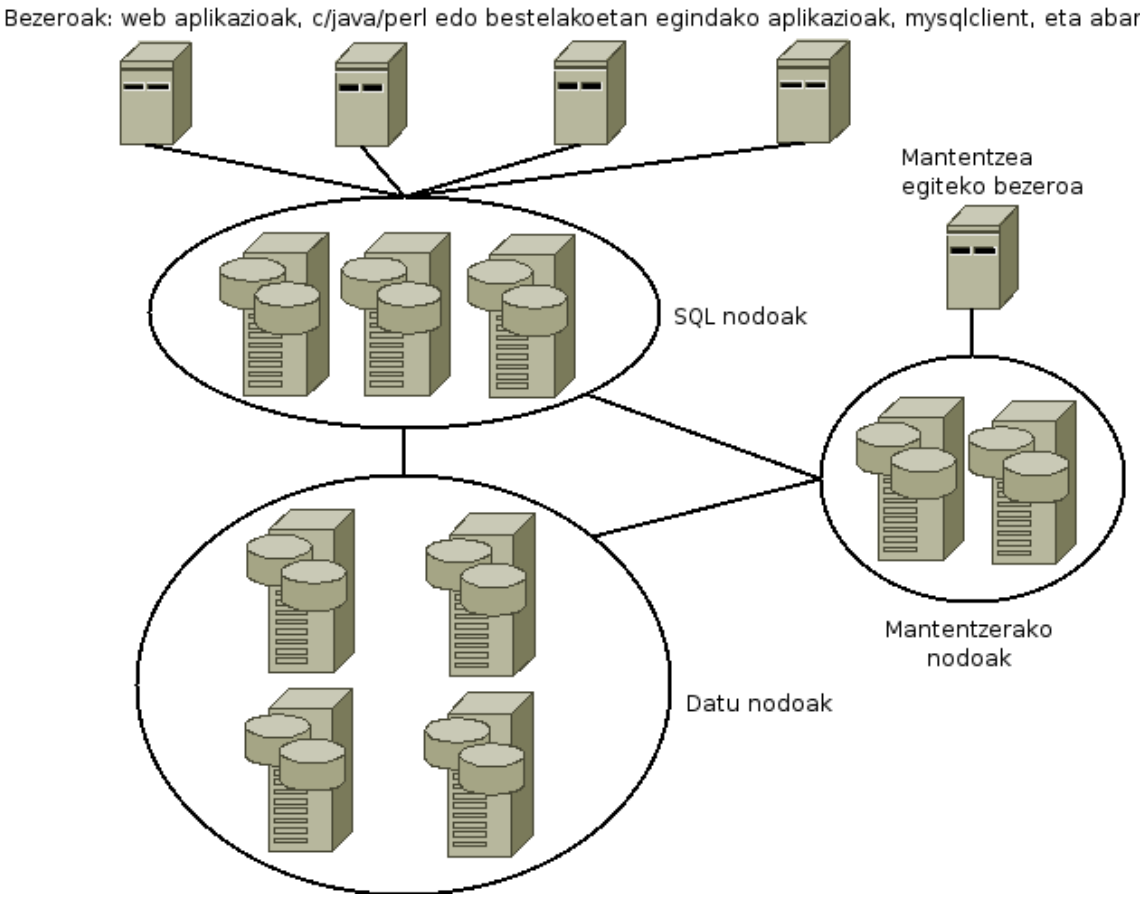
Eta honekin nahikoa da oinarrizko erreplikazioa egiteko. Nagusian sortutako datu-baseak, taulak eta gainontzekoak automatikoki sortuko dira morroian ere.

6.1.3.2.- Cluster sistema Mysql erabiliz

Ikusi dugunez, nagusi/morroi sistema bat inplementatzea oso sinplea da *Mysql* erabiliz. Ingurune askotarako nahikoa bada ere, *cluster*-a erabili beharko dugu idazketa kopuruak nagusiaren ahalmena gainditutakoan, edo aplikazioak ezin direnean aldatu idazketa eta irakurketen artean bereizketa egiteko.

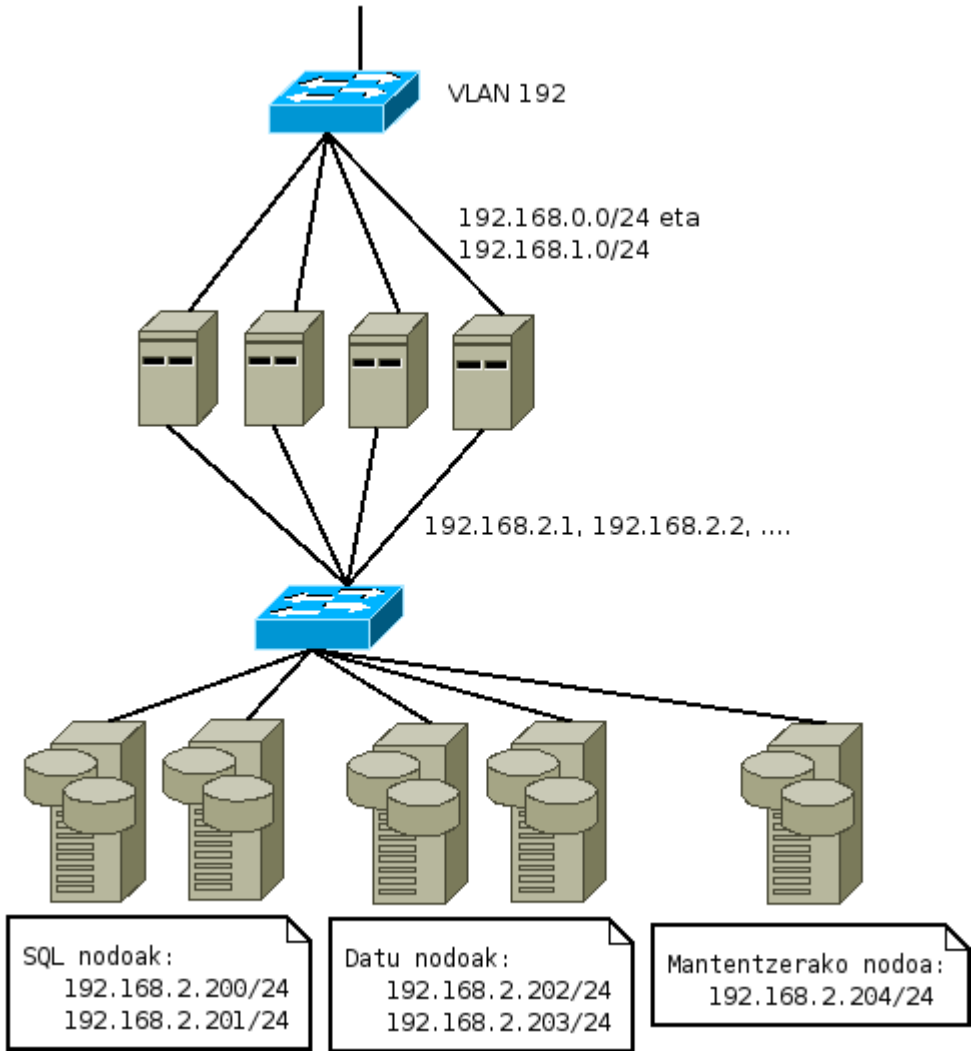
Mysql bidez egindako *cluster* batek osagai hauek ditu:

- Datuetarako nodoa: Honen ardura datuen biltegiratzea da. *Mysql*-ren 4.1 eta 5.0 bertsioek datu hauek memorian gordetzen dituztenez, gomendagarria da zerbitzari mota honetan memoria fisiko asko egotea. Hauetako nodo bat hasieratzeko *ndbd* komandoa egikaritu behar da.
- SQL nodoa: Hauen ardura datuetarako nodoak atzitzea da. Nolabait esateko, ohiko *Mysql* zerbitzari arruntak dira, baina datuak jasotzeko datu-nodoak erabiltzen dituztenak. SQL nodo guztiek izango dute irakurketak eta idazketak egiteko ahalmena. Nodo hauek ohiko *mysqld* komandoaz egikaritzen dira, baina *ndbcluster* aukerarekin.
- Mantentzerako nodoa: Hauen ardura datu eta SQL nodoen kontrola eramatea da. Nodoen konfigurazio, hasieratze eta gelditzea edo segurtasun kopiak egitea nodo mota honen esku geldituko da. Hasieratzeko *ndb_mgmd* komandoa erabiltzen da.



Irudia 25: Mysql cluster-a

Irudian agertzen den makina kopurua adibide bat baino ez da. Erreplikazio minimoa izateko SQL eta mantentzerako nodo bana eta datu nodo bi behar dira. Fnneten bina SQL eta datu nodo jarriko dira, eta mantentzerako bat.



Irudia 26: Mysql cluster-a Fnneten azpiegituran

Cluster honen inplementazioarekin hasi aurretik, gogoratu konfigurazio lerro interesgarrienak baino ez ditudala idatziko.

Hasteko */etc/hosts* fitxategiak sortuko ditut *cluster*-aren makina guztietan.

```
# /etc/hosts, clusterreko makina guztiekin
192.168.2.200  sql1
192.168.2.201  sql2
192.168.2.202  datu1
192.168.2.203  datu2
192.168.2.204  mantentzea1
```

Hau eginda, hasi gaitezen SQL nodoak konfiguratzen, */etc/mysql/my.cnf* fitxategian[70]:

```
[MYSQLD]
ndbcluster
ndb-connectstring=192.168.2.204     # edo mantentzea1
```

Bi SQL makinetan berdina jarri behar da. *nbdcluster* aukerarekin *cluster*-a sortzeko prestatuko dugu zerbitzaria, eta *ndb-connectstring* aukerarekin mantentzerako zerbitzaria non dagoen esango dugu.

Segi dezagun datu nodoekin, */etc/mysql/my.cnf* fitxategian (bi makinetan berdina):

```
[MYSQL_CLUSTER]
ndb-connectstring=192.168.2.204     # edo mantentzea1
```

Eta bukatzeko, mantentzerako nodoa konfiguratu beharko dugu, */etc/mysql/config.ini* fitxategian[71]:

```
[NDBD DEFAULT]
NoOfReplicas= 2
DataMemory=20M
IndexMemory=10M
DataDir= /var/lib/mysql-cluster

[NDB_MGMD]
Hostname= mantentzea1
DataDir= /var/lib/mysql-cluster

[NDBD]
HostName= datu1
```

70 Konfigurazio fitxategi hau *Debian testing* ingurunetan erabiltzen da.

71 Nahi dugun tokian jarri dezakegu fitxategi hau, eta nahi dugun izenarekin, baina *config.ini* da gehienetan erabiltzen dena.

```
[NDBD]
HostName= datu2

[MYSQLD]
HostName= sql1
[MYSQLD]
HostName= sql2
```

Ikusten denez, fitxategi honetan izenak erabili ditut, eta ez IP helbideak. Honekin batera aukera batzuk jarri ditut egin daitekeenaren ideia bat izateko.

Zerbitzari bakoitza hasieratzeko prozesua Linux banaketa bakoitzean desberdina da, baina funtsean komando hauek exekutatu behar dira:

```
# Mantentzerako nodoan.
ndb_mgmd -f /etc/mysql/config.ini

# Datu nodo bakoitzean, eta konfigurazio fitxategi lehenetsia
# /etc/mysql/my.cnf dela kontutan hartuz.
ndbd        # exekutatzen den lehenengo aldian ndbd --initial

# SQL nodo bakoitzean, /etc/mysql/my.cnf erabiliz:
mysqld
```

Hasieratzen den lehenengo zerbitzaria mantentzerako nodoa dela izan behar da kontutan. Behin makina guztiak hasieratu ditugunean, *ndb_mgm* mantentzerako bezeroa exekutatzen badugu, eta *show* komandoa idatzita:

```
root@makina:/# ndb_mgm –ndb-mgmd-host=192.168.2.204
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: 192.168.2.204:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.2.202  (Version: 5.0.32, Nodegroup: 0, Master)
id=3    @192.168.2.203  (Version: 5.0.32, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.2.204  (Version: 5.0.32)
```

```
[mysqld(API)]   2 node(s)
id=4    @192.168.2.200  (Version: 5.0.32)
id=5    @192.168.2.201  (Version: 5.0.32)

ndb_mgm>
```

Ikusten denez *cluster*-aren osagai guztiak arazorik gabe detektatu dira. Hemendik aurrera gure datu-basetako taulak *cluster*-ean sartzeko aukera izango dugu, bai *sql1* edo *sql2* nodoak erabiliz. Adibidez, demagun "clusterdatubasea" izeneko datu-base bat dugula, eta "taula" izeneko taula bat sortu nahi dugula:

```
root@makina:/# mysql -u erabiltzailea -h 192.168.2.200 -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 14
Server version: 5.0.32-Debian_7etch1 Debian etch distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use clusterdatubasea;
Database changed
mysql> CREATE TABLE `taula` (
    ->   `id` int(11) NOT NULL auto_increment,
    ->   `mota1` char(35) NOT NULL default '',
    ->   `mota2` char(20) NOT NULL default '',
    ->   `mota3` int(11) NOT NULL default '0',
    ->   PRIMARY KEY  (`id`)
    -> ) ENGINE=NDBCLUSTER DEFAULT CHARSET=latin1;
Query OK, 0 rows affected (1.34 sec)
```

192.168.2.200 edo 192.168.2.201 zerbitzariak erabil daitezke inongo arazorik gabe, adibide honetan 192.168.2.200 erabili badut ere. Taula *cluster*-ean sartzeko *engine=ndbcluster* aukera erabili behar da. Hemendik aurrera, bai 192.168.2.200 bai 192.168.2.201 zerbitzarietan "taula" taularen gainean egindako eragiketak beste zerbitzarian ia denbora errealean ikusiko dira.

Hasierako konfigurazio hau hartuta, hurrengo urratsa *cluster* sistemaren konfigurazioa optimizatzea litzateke, *buffer*, *log*, *cache* eta beste hainbat aukera gure aplikazioari eta makinen ahalmenari egokituz, baina hau proiektu honen esparrutik kanpo gelditzen da. Konfigurazio aukera guztiak zeintzuk diren jakiteko (Mysql, 2006) web gunea erabil daiteke.

6.2.- Datuen biltegiratzea. Fitxategi sistemak

Karga banatzeko sistemek sortzen duten buruhausterik handienetakoa edukiaren biltegiratzearekin dator. Demagun PHP bidez egindako web aplikazio bat garatu dugula bezeroek liburuen katalogo bat ikus dezaten. Katalogoaren informazioa datu-basean gordetzen da, eta irudiak web zerbitzariak atzitu dezakeen direktorio batean. Honekin batera, aplikazioak katalogoa administratzeko atal bat ere badu, datu-basean informazio berria gehitu eta irudi berriak igotzeko aukerarekin.

Demagun web aplikazioaren administratzaileak liburu bati buruzko informazioa gehitzen duela (PHP aplikazioa erabiliz), inongo arazorik gabe, baina katalogoa egiaztatu nahi duenean, liburu berria katalogoan agertzen bada ere, honen irudia ez da beti ikusten. Noski, administraziorako aplikazioak zerbitzari erreal baten kontra egiten du lan, eta honek esan nahi du irudia zerbitzari bakarrera igo dela. Administratzaileak katalogoa egiaztatzen duenean, fitxategia igo eta minutu batzuetara, karga banatzeko sistemak beste zerbitzari erreal batera eraman du trafikoa, datu-basearen informazioa espero bezala jasoz, baina irudirik gabe.

Honetarako konponbide bila hasita, aurreneko ideia edukia zerbitzari guztietan kopiatzea da, eskuz edo prozesu automatiko baten bitartez. Hau egiteko tresna interesgarri asko dago. Adibide bat ematearren, *Rsync* erabiliz zerbitzari erreal batean dagoen edukia oso erraz kopiatu daiteke beste guztietara, eta oso modu eraginkorrean.

Esan beharrik ez dago honelako kopia asinkronoak ezin direla askotan erabili. Zerbitzari errealen kopurua handia denean, edo edukia oso dinamikoa denean, bi kasu aipatzearren, ezingo da honelako sistemarik erabili. Zerbitzari batean aldatutakoa beste guztietan denbora errealean ikusi behar denean, orduan beste konponbide bat bilatzeko ordua da. Hau, gaur egun, bi modu nagusitan egin daiteke:

- NAS sistema baten bidez. Hitz gutxitan, zerbitzari batean eduki guztia gorde, eta gero NFS edo SMB/CIFS protokolo ezagunen bitartez zerbitzari errealentzat eskuragarri jarri. Honen abantaiarik nagusienak bi direla esango nuke; batetik software eta hardware estandarra erabiltzen dela, batez ere NFS (edo SMB/CIFS) zerbitzaria Linux bidez inplementatzen bada, eta bestetik oso teknologia ezaguna dela, NFS, TCP/IP eta *Ethernet* bezalakoak dagoeneko urte askotan zehar erabili direlako. Kontrakoak aipatzeko, garrantzitsuena, zalantzarik gabe, NFS zerbitzariarekiko sortzen den menpekotasuna da. Honelako sistema batean zerbitzariaren matxurak azpiegitura guztian du eragina.
- SAN sistema bat erabiliz[72]. Azken urteotan teknologia honetan oinarritutako biltegiratzerako azpiegiturek gorakada handia izan dute. iSCSI[73] teknologian oinarritutakoak badaude ere, gehienetan zuntza erabiltzen da abiadura azkarrak lortzeko. Honekin batera, biltegiratzeko ahalmen handia eskaintzen dute. SAN teknologiaren mamia biltegiratzerako bereziki sortutako sare bat da, non gehienetan SCSI komandoak igarotzen diren, bai zuntzean bai IP gainean kapsulatuta. Biltegiratze ahalmen handia, azkarra, hedagarria eta fidagarria nahi bada, SAN da aukerarik egokiena. Hori bai, guzti hauek lortzeko aurrekontu handia behar da.

Fnnet enpresarentzat NAS sistema bat inplementatuko dugu, baina zerbitzari bakarra erabili ordez bi erabiliko ditugu, fidagarritasun mailahandituz.

72 SAN teknologiaren inguruan liburuak, aldizkariak, karrera bukaerako proiektuak eta abar idatzi dira. http://en.wikipedia.org/wiki/Storage_Area_Network helbidean oinarrizko deskribapena irakur daiteke, eta http://www.redbooks.ibm.com/redbooks/pdfs/sg245470.pdf helbidean zerbait sakonagoa.

73 http://en.wikipedia.org/wiki/ISCSI helbidean sarrera txiki bat eta zenbait esteka aurki daitezke.

6.2.1.- Oinarrizko inplementazioa

Fnnetentzat SAN sistema fidagarri bat eraikitzea gehiegizko inbertsioa da. Eraginkortasun beharrei begira ere ez da *Ethernet*-ek ematen duena baino gehiago beharko denik aurreikusten; ez behintzat hasierako urtetan. Gainera, hardware eta softwareari begira ere azpiegitura homogeneoa nahi da ahal den heinean. Hau honela, eraginkortasun arazorik ez dagoen bitartean NAS sistema bat inplementatzea aukeratu da. Hori bai, zerbitzari bakarra izateak eragiten duen menpekotasuna onartezina da.

Fidagarritasun maila igotzeko NFS zerbitzarian dagoen edukia gutxienez beste makina batean gorde beharko genuke. Bigarren zerbitzari hau gauza izan behar da edozein momentutan lehenengoaren tokia hartzeko daturik galdu gabe. Fnneteentzat inongo daturik ez galtzea da lehentasunik handiena, baina denborarekin inongo etenaldirik ez izatea ere nahi da, noski.

Behar hauek guztiak DRBD deritzion softwarearekin bideratuko ditugu. Ideia nagusia dagoeneko ezaguna dugu; makina batek izango du nagusiaren papera, eta bigarren makina batek jasoko ditu, *Ethernet* sare arruntaren bitartez, lehenengo honetan egindako idazketak. DRBDren 8.0.0 bertsioaz geroztik bi makinak nagusi bezala aritzeko aukera dago, *cluster* moduan. Hau egiteko GFS (Gfs, 2006) moduko fitxategi sistema bat behar da[74].

Oraingoz, nagusiak irakurketa eta idazketak egiteko ardura izango du; bigarren zerbitzaria kopiak egiteko baino ez dugu erabiliko. Honek esan nahi du nagusiaren matxuraren aurrean zerbitzari errealak eskuz edo *script* baten bitartez aldatu beharko genituzkeela, muntaketa-puntuak *backup*-erako makinara aldatuz. Honek MTTR balioa handitzen du, baina oraingoz sinpletasunari emango diogu lehentasuna. Hurrengo atalean honen aurrean egin daitezkeenak aipatuko ditut.

74 GFS *cluster*-etan aritzeko bereziki sortu zen. Hemendik aurrera nagusi/morroi diseinuan oinarrituko naiz, non ia edozein fitxategi sistema erabili daitekeen.

DRBD instalatzeko pausoak dagoeneko ezagunak ditugu. Batetik kernela konpilatu behar da DRBD dispositiboak sortu ahal izateko. Hau egin eta gero aplikazio txiki batzuen bitartez konfiguratuko da.

Linux kernelaren iturburu kodea */usr/src/linux* direktorioan badago, eta *drbd-8.0.2.tar.gz*[75] */usr/src/* direktorian deskomprimituta, kernel modulua eta aplikazioak sortzeko urratsak hauek dira:

```
cd /usr/src/drbd-8.0.2
make all install

#DRBD dispositiboak /dev/drbd[0-9] itxurakoak dira. Gure sistemaren
#konfigurazoaren arabera automatikoki edo eskuz sortu beharko ditugu
#Script txiki bat eskuz sortzeko:
for i in `seq 0 15` ; do
    test -b /dev/drbd$i || mknod -m 0660 /dev/drbd$i b 147 $i;
done
```

Modulu bezala eraiki dut hemen, baina nahi izanez gero kodea kernelean zuzenean gehitzeko aukera dago. Edozein dela gure aukera, ziurtatu behar dugu modulua edo aldatutako kernela sistema hasieratzen denean kargatuko dela. Informazio gehiago *tar.gz* fitxategiaren barruan dagoen INSTALL fitxategian irakur daiteke.

Aplikazioaren konfigurazioarekin hasi baino lehen biltegiratzea egiteko sareko azpiegitura zein izango den erabaki behar da. Zerbitzari erreal eta datu-base kudeaketa-sistemen arteko komunikaziorako egin bezalaxe, hemen ere sare isolatu bat sortu genezake inongo arazorik gabe, *switch* berri edo VLAN bitartez, baina oraingoz azpisare berdina erabiliko dugu. Nagusiak 192.168.2.220 helbidea izango du, eta *backup*-erako makinak 192.168.2.221.

Sistemaren oinarrizko konfigurazioa */etc/drbd.conf* fitxategian egiten da, Honekin batera *drbdadm* aplikazioa ere erabiltzen da. Oinarrizko *drbd.conf* fitxategi bat hau izan daiteke (nagusi eta *backup* makinetan berdina):

75 Momentuko azken bertsioa da, DRBDren web gunetik eskuragarria.

```
global {
    usage-count yes;
}

common {
  # rate balioaren unitatea: byte
  syncer {
    rate 10M;
  }

  protocol C;

  handlers {
    pri-on-incon-degr "echo o > /proc/sysrq-trigger ; halt -f";
    pri-lost-after-sb "echo o > /proc/sysrq-trigger ; halt -f";
    local-io-error "echo o > /proc/sysrq-trigger ; halt -f";
    outdate-peer "/usr/sbin/drbd-peer-outdater";
  }

  startup {
    wfc-timeout  300;
    degr-wfc-timeout 120;    # 2 minutu
  }

  disk {
    on-io-error   detach;
  }

  net {
    timeout       60;    #  6 segundo
    connect-int   10;    # 10 segundo
    ping-int      10;    # 10 segundo
    ping-timeout   5;    # 500 ms

    after-sb-0pri disconnect;
    after-sb-1pri disconnect;
    after-sb-2pri disconnect;
    rr-conflict disconnect;
  }
}
```

```
resource drbd-baliabidea {
   # Nagusia 192.168.2.200 makinaren izena da.
   on nagusia {
     device     /dev/drbd0;
     disk       /dev/sda1;
     address    192.168.2.200:7788;
     flexible-meta-disk internal;
   }

   # Backup 192.168.2.201 makinaren izena da.
   on backup {
     device     /dev/drbd0;
     disk       /dev/sda1;
     address    192.168.2.201:7788;
     flexible-meta-disk internal;
   }
}
```

Konfigurazio hau erabilita, muntaketa-puntua */dev/drbd0* dispositiboa izango da bi makinetan. DRBDren lana izango da egindako idazketak */dev/sda1* disko gogor lokalaren partizioan eta *backup* makinan egitea.

Hasteko DRBDren meta-datuak hasieratu behar dira. Gure adibidean bloke dispositibo berdina erabili dugu hau gordetzeko *flexible-meta-disk internal* aukerarekin. Bi makinetan hasieratzen denean DRBD martxan jar genezake; *Debian testing* banaketan */etc/init.d/dbrd start script*-a erabiliz:

```
drbdadm create-md drbd-baliabidea

# Sistema ondo hasieratu bada "/proc/drbd" fitxategiaren irteera:
root@nagusia:/# cat /proc/drbd
version: 8.0.2 (api:86/proto:86)
SVN Revision: 2844 build by root@mordor, 2007-04-15 20:51:57
0: cs:Connected st:Primary/Secondary ds:UpToDate/UpToDate C r---
 ns:1048576 nr:0 dw:0 dr:1048576 al:0 bm:64 lo:0 pe:0 ua:0 ap:0
  resync: used:0/31 hits:65472 misses:64 starving:0 dirty:0 changed:64
  act_log: used:0/127 hits:0 misses:0 starving:0 dirty:0 changed:0
```

Hauek eginda, prest gaude fitxategi sistemak sortzeko[76]. Adibide gisa, nagusia izango den makinan *Reiserfs* motako fitxategi sistema sortuko dugu, eta gero fitxategi bat sortuko dugu.

```
root@nagusia:/# mkreiserfs /dev/drbd0
root@nagusia:/# mount /dev/drbd0 /drbd_edukia
root@nagusia:/# echo "fitxategi proba bat" > proba1.txt
```

Nagusian arazorik balego, batetik *backup* makina *secondary* egoeratik *primary* egoerara pasa (*drbdadm* erabiliz), eta bestetik */dev/drbd0 /drbd_edukia* direktorioan muntatuko genuke. Noski, *proba1.txt* fitxategia hor egongo litzateke.

Oinarri honetatik abiatuta, NFS edo SMB/CIFS zerbitzari bat erraz konfiguratu ahal izango dugu, kontutan izanda */drbd_edukia* izango dela bezeroentzat eskuragarri jarritako edukia.

Eta honekin DRBDren oinarrizko inplementazioa egin da. Hurrengo atalean biltegitatze sistema honen fidagarritasun maila nola igo daitekeen ikusiko dugu (MTTR balioa txikituz).

6.2.2.- Fidagarritasunaren hobekuntza

Ikusi denez, biltegiratzeari fidagarritasun maila minimo bat emateak ez dio sistema osoari aparteko zailtasunik gehitu. *Backup* makina bakar bat gehitu dugu, oinarrizko NFS zerbitzariaren konfigurazio ia berdinarekin. Mantentzeari begira ere, makina bakar batek ematen duen lan berdintsua izango dugu. Arazoa berreskuratze prozesu bat aurrera eraman behar denean dator; hots, NFS zerbitzari nagusia hondatuko balitz, *backup* makina *primary* egoerara pasa beharko genuke, */dev/drbd0 /drbd_edukia* direktorioan

76 Egia esateko, zenbait urrats kendu ditut emandako azalpen honetan. Izatez, DRBD lehenengo aldiz exekutatzen denean hasierako sinkronizazio bat egin behar da nagusian. DRBDren bertsioaren menpe dago hau, baina normalean *drbdadm -- --overwrite-data-of-pær primary all* da egikaritzen den komandoa. Azalpena argiago izateko, esan bezala, pausu hau kendu dut.

muntatu, *backup* NFS zerbitzaria martxan jarri, eta azkenik zerbitzari erreal guztiak eskuz aldatu beharko genituzke, 192.168.2.220 makinan dauden muntaia-puntuak 192.168.2.221 makinara aldatuz[77]. Hau automatikoki, *script* bitartez, egin badaiteke ere, MTTR balioarentzat kaltegarria da, batez ere *script*-a exekutatu arte igaro behar den denbora altua denean. Ingurune askotan denbora hau onargarria izan daiteke, baina ez guztietan.

Konpondu beharreko arazo nagusia NFS bezeroek (zerbitzari errealak) duten muntaketa-puntuan dago. IP helbide birtual bat erabiliko dugu, noski, orain arte askotan egin dugun bezalaxe, baina VRRP erabili ordez beste aplikazio erabilgarri bat aurkeztuko dut hemen, Linux ingurunetan *Heartbeat* izenarekin ezagutzen dena[78] (Linux-ha, 2005). Helbide birtualak maneiatzeaz gain, *Apache*, *Sendmail*, suhesi erregelak, DRBD edo ia beste edozer kudeatzeko gauza da aplikazio hau. Honela, *Heartbeat* bidez kontrolatutako makinetako batean arazoren bat balego, beste batek hartuko luke bai helbidearen bai konfiguratutako aplikazioak exekutatzearen ardura.

Fnneten kasuan, *Heartbeat* softwarea bi NFS makinetan instalatuko dugu. Bere ardura nagusia IP helbide birtualaren eta momentuan exekutatzen egongo den NFS zerbitzariaren kontrola izatea da. Funtzionamendu egoera arruntean makina batek helbide birtuala, DRBD nagusiaren rola eta NFS zerbitzaria izango ditu, baina *backup*-ak arazoren bat detektatuko balu bere gain hartuko luke bai helbidea, bai DRBD, bai zerbitzaria exekutatzearen ardura. Guzti hau bezeroentzat gardena izango litzateke.

Hasteko, *Heartbeat* aplikazioa instalatzeko *Debian* ingurunetan izen berdineko bi pakete daude eskuragarri: *Heartbeat* eta *Heartbeat-2*. Azkenekoa aurreratuagoa da, ahatsuagoa, baina hurrengo lerroetan azalduko dudana egiteko biak dira erabilgarri. Aplikazioaren konfigurazio osoa (Linux-ha, 2005) web gunean dagoen dokumentazioa jarraituz egin

77 *Backup* makinan 192.168.2.220 helbidea jartzeko aukera ere badago, baina edozein kasutan bezeroen muntaketa berriz egin beharko litzateke.

78 Izatez heartbeat linux-ha aplikazio multzoaren osagai bat baino ez da.

daiteke. Hemen ideia orokorra zein den ikusteko konfigurazio zati txiki bat azalduko dut. Kode zati hau bai nagusian bai *backup*-ean idatzi behar da, normalean */etc/ha.d/haresources* fitxategian:

```
nagusia IPaddr::192.168.2.222/24/eth0 drbddisk::drbd-baliabidea \
Filesystem::/dev/drbd0::/drbd_edukia::reiserfs nfs-common nfs-kernel-
server
```

Non:

- *nagusia*, NFS zerbitzari nagusiaren izena da.
- *Ipaddr::192.168.2.222/24/eth0* IP helbide birtuala da, 24 maskararekin, eta *eth0* sare interfazean. Hau izango da bezeroek beti erabiliko duten helbidea.
- *drbddisk, Heartbeat* aplikazioaren *script* bat da. Bere ardura *drbd-baliabidea* nagusi edo morroi bezala markatzea da.
- *Filesystem* beste *script* bat da. Emandako datuekin fitxategi sistema muntatuko du.
- *nfs-common* eta *nfs-kernel-server*, *Debian* banaketak */etc/init.d* direktorioan NFS zerbitzuak martxan jartzeko dituen *script*-ak dira.

Beste hitz batzuetan, *nagusia* izeneko makinak izango du helbide birtuala eta DRBD disko nagusi bezala, konfiguratutako tokian muntatuta. NFS zerbitzaria ere aktibatuta egongo da. Baina nagusian arazorik balego, *Heartbeat* berak IP helbidea morroira eramango luke, DRBD sistema egokitu, eta NFS zerbitzaria martxan jarriko luke.

DRBD, *Heartbeat* edo NFS zerbitzariaren bertsioen arabera inplementazioa modu batean edo bestean egingo da. Hemen oso azaleko hurbilpen bat baino ez dut egin. Gaian gehiago sakontzeko aplikazio desberdinen web gunetan dokumentazioa dago eskuragarri[79].

79 Tutorialik ere badago. Bi aipatzearren:

http://www.howtoforge.com/high_availability_nfs_drbd_heartbeat

http://www.linux-ha.org/DRBD/NFS

7.- MONITORIZAZIO ETA SEGURTASUNAREN GARAPENA

Enpresa batean, sare azpiegituraren osagaietako bat gelditzen denean egoera gogorrak ematen dira. Hardware arazoak, sistema eragileen blokeaketak, datu-base kudeaketa-sistemen ahalmenaren gainditzea edo beste edozer ematen denean berehala izango dugu norbait gure atzean gertatu denaz galdezka. Hauen aurrean erantzunik ez izatea baino gauza txarragorik ez dago.

Segurtasunak ere badu monitorizazioarekin zer ikusirik. Suhesien moduko tresnen garrantzia azpimarratu beharrik ez dagoela iruditzen zait, oso gureganatuta dugu hauen beharra, eta proiektu honetan zehar oinarrizko suhesiak eraikitzeko softwarea dagoeneko aipatu dut. Baina tresna hauekin bakarrik askotan ezin da azpiegitura segurua denik ziurtatu, are eta gehiago gure sarea irekia denean, esate baterako web zerbitzu bat duelako. Gauza al da gure suhesia erabiltzen dugun web aplikazioaren *cross-site scripting*, SQL *injection* edo *code injection*[80] arazoen aurrean trafikoa galerazteko? Eta DNS trafiko bezala ezkutatuta gure bezeroen datu pribatuak bidaliko balira[81]? Orokorrean hitzeginda, zer egin gure suhesi sistemak erasoren bat ezin duenean detektatu? Suhesirik onenak ere arazoak izan ditzakeelako, suhesi perfekturik ez dagoelako, gure azpiegituraren segurtasunean pauso bat aurrera eman behar dugu, NSM[82] monitorizazio teknikekin.

Atal honetan bai NSM bai azpiegituraren baliabideen monitorizazioa egiteko zenbait ideia aurkeztuko ditut, software librean oinarritutako tresnekin, eta Fnneten inguruneari egokituta.

80 http://en.wikipedia.org/wiki/Cross_site_scripting, http://en.wikipedia.org/wiki/SQL_Injection, http://en.wikipedia.org/wiki/Code_injection.

81 Bigarren galderaren erantzuna http://l7-filter.sourceforge.net/ moduko softwareak eman dezake.

82 Network Security Monitoring

7.1.- Monitorizazioa

Sare azpiegitura bat mantentzeko, kontrolatuta izateko, monitorizazio egokia ezinbestekoa da. Batetik, sistemen administrazioaren ikuspuntutik tresna aparta delako (eta derrigorrezkoa esango nuke), eta bestetik sarearen egoeraren ikuspuntu intuitiboagoa emateko, bai bezeroen aurrean, bai enpresaburuen aurrean.

Ez da erraza monitorizatu ezin daitekeen zerbait aurkitzea. Izatez, enpresa batean guztia monitorizatzeko baliabiderik ez izatea ohikoagoa da, batez ere kontutan badugu emaitzak interpretatzea langile adituen esku dagoela. Azterketa automatikoa erabiltzen bada ere, batez ere azpiegitura handietan, azkenean langileek ordu asko sartu behar izaten dituzte emaitza hauek edo irudien analisia egiten, eta hau da, hain zuzen ere, monitorizazioaren arazorik handiena; sistemak ezartzen direla, baina denborarekin askotan albo batean ahaztuta bukatzen dutela.

Gure azpiegituran monitorizatu[83] beharreko arloak, orokorrean hartuta, hurrengo zerrendan ikus daitezke:

- Zein da sarearen erabilpena, bai osotasunean bai atalka? Zein da denboran zehar ikusten den joera?
- Sarean edo honen zatiren batean, momenturen batean, ahalmen mugatik gertu gaude? Zein da joera?
- Sarearen errendimenduak, osotasunean edo atalen batean, momenturen batean, behera egiten al du?
- Sarearen erantzun deborak inoiz onartezinak izaten al dira?
- Sarean arazorik eman al da? Non hasi da? Noiz? Noiz konpondu da? Noiz berreskuratu da egoera arrunta?
- Sarearen atalen batek errore tasa altuegia[84] al du?

83 Gogoratu atal honetan ez naizela segurtasunaren monitorizazioaz ari.

84 Ingurune askotan eta zenbait gailutan errore kopuru txikia onargarria izaten da.

Berdintsua egin daiteke, esate baterako, web zerbitzariak monitorizatzeko:

- Zenbat web eskaera zerbitzatzen dira? Zein da denboran zehar ikusten den joera?
- Zerbitzariaren ahalmen mugatik gertu al gaude? Joeraren arabera, noiz iritsiko gara? Gabonak bezalako garaietan eman daitekeen gorakada jasotzeko moduan gaude?
- Zenbat denbora behar da eskaera bat zerbitzatzeko? Balio hau egonkorra da edo lan kargarekin batera gehiegi egiten al du gora?
- Zein da zerbitzariaren memoriaren erabilpena? Zenbat sarrera/irteera eragiketa ditu? Sare interfazeen erabilpena? Prozesurako ahalmenaren erabilpena?
- Zerbitzarian arazorik eman al da? Non hasi da? Noiz? Noiz konpondu da? Noiz berreskuratu da egoera arrunta?

Zerrenda hau ingurune bakoitzari egokitu eta gero, aholkularitza lanaren hurrengo pausoa tresnen aukeraketa egitea litzateke. Software libreak katalogo zabala eskaintzen du, baina gehienetan, eta ondorioz baita Fnneten kasuan ere, bi motatako softwarea behar dela ondorioztatzen da.

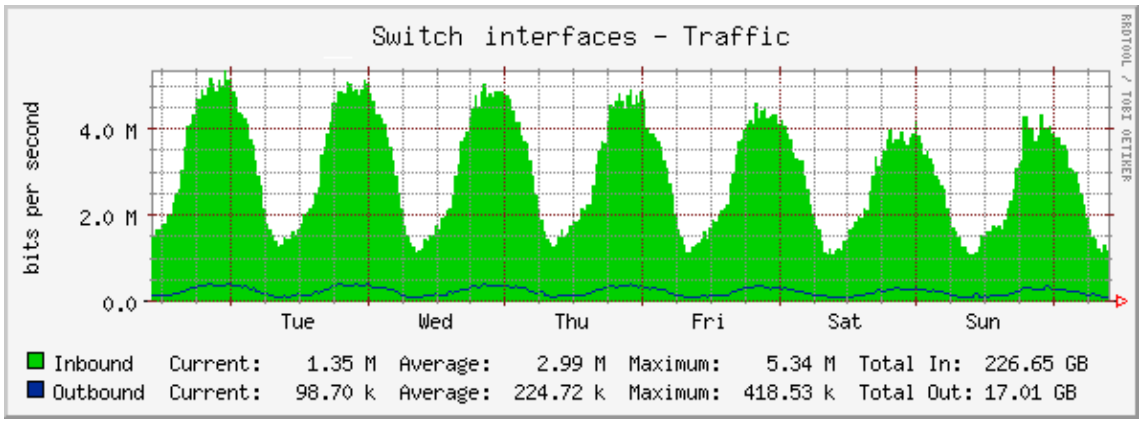
1. Zerbitzuak martxan dauden edo ez ikusteko aplikazioa. Bere ardura nagusia izango da gailuak eta monitorizatutako aplikazioak (Web zerbitzariak, *Mysql*, SMTP zerbitzariak, ...) behar bezala funtzionatzen duten aztertzea. Erantzun denbora altuegiak edo komunikazioa galduko balitz monitorizazio sistemak berak posta elektroniko, SMS edo beste tresna baten bidez abisatuko liguke. Guzti hau egiten duten tresnen artean *Nagios* da, seguraski, gehien erabiltzen dena.
2. Denboran zehar azpiegituraren atal guztien funtzionamenduaren adierazpen grafikoa emango duen aplikazioa. Erabilitako banda zabalera, web eskaerak zerbitzatzeko behar den denbora, *Mysql* zerbitzarian egindako *select* eragiketak edo jasotako mezu elektronikoen kopurua honen adibide batzuk baino ez dira. Izatez, zenbaki batekin neurtu daitekeen edozer eraman daiteke grafiko batera. Irudi hauekin lortu nahi duguna da, batetik funtzionamendu patroi batzuk izatea[85],

85 Hilabetetan zehar egindako grafikoekin asteko egun bakoitzean, normalean, zenbat web eskaera izaten

eta bestetik denboran zehar ematen ari den joera[86] ikustea. Guzti hau praktikara eramateko aukera asko dago; gehienetan datuak jasotzeko SNMP protokoloa[87] erabiltzen da, eta irudiak egiteko MRTG/*RRDtool*, *Cacti* bezalako web interfaze batekin (4.1.7.2 atala) .

Inplementazio zehatza baino irudien adibide batzuk jarriko ditut hemen (guztiak *RRDtool* bidez eginak):

Aste batean zehar *switch* interfaze batean emandako banda zabaleraren erabilpena:



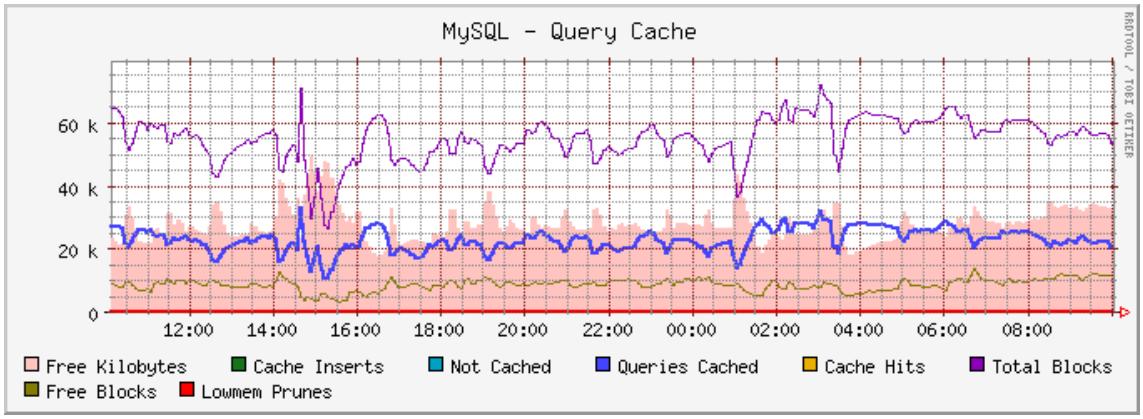
Irudia 27: Banda zabalera switch baten interfaze batean

Mysql zerbitzariekin erlazionatutakoak, *cache*-ari buruzko informazioa eta egindako SQL komandoen datuak:

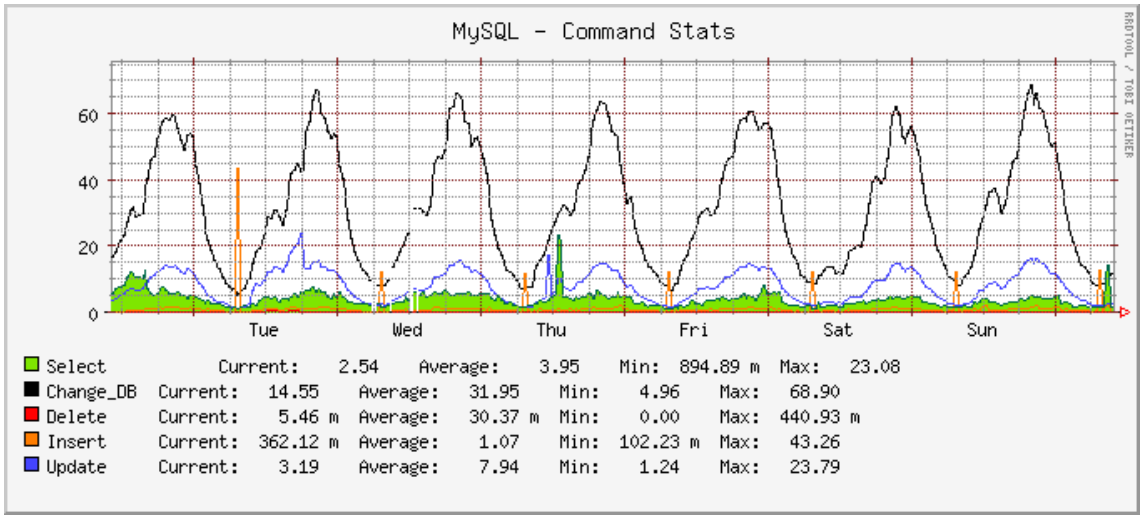
dugun jakingo dugu. Honela, egun arrunt batean izaten duguna baino askoz gehiago edo gutxiago ikusiko bagenu, zerbait gertatzen ari dela ondorioztatu genezake.

86 Adibidez, trafikoa hilabete bakoitzean ehuneko bost igotzen dela.

87 SNMPren inplementaziorik erabiliena http://net-snmp.sourceforge.net/ helbidekoa da.



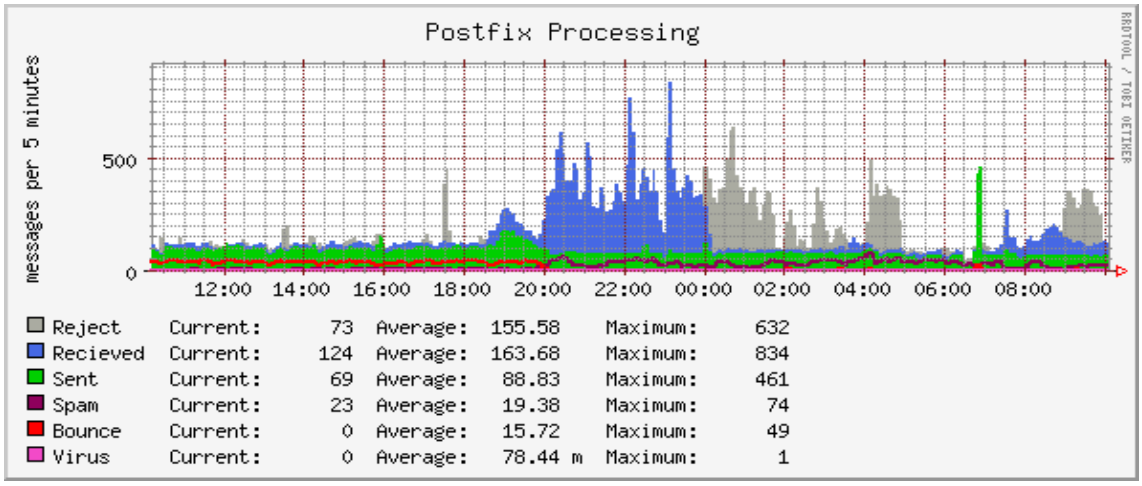
Irudia 28: Mysql cache-aren erabilpena



Irudia 29: Mysql egindako eragiketak

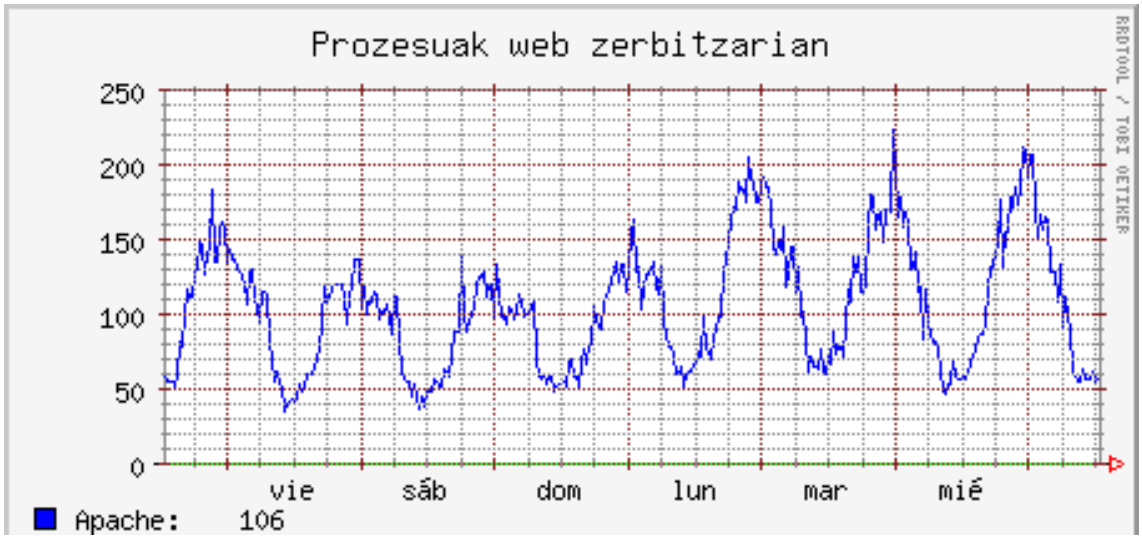
Ikusi astearteko *insert* komandoen gorakada. Gainontzeko egunetan ere egoten da, baina txikiagoa. Fnneten kasuan analisiak esango luke arazorik egon den edo astearte horretan katalogoan normalean baino datu gehiago sartu den.

Posta zerbitzariekin ere informazio asko eskuratu daiteke, adibidez:



Irudia 30: Posta zerbitzariak jaso eta bidalitako mezuen informazioa

Hurrengo irudian zerbitzari batean exekutatzen dagoen *Apache* prozesuen kopurua ikusten da:



Irudia 31: Zerbitzari baten prozesuen kopurua

Hauek adibide batzuk baino ez dira. Guzti hau erabilgarria izateko Fnnetek monitorizazioaren beharra bereganatu behar du, astean zehar zenbait lanordu lortutako emaitzen analisiarako erabiliz. Hau lortzen ez bada, irudien azterketarik egiten ez bada, sistemarik perfektuena ere hutsaren hurrengoan geldituko da oso denbora gutxian.

7.2.- Segurtasuna

Sareko azpiegitura baten segurtasunaz hitz egiten denean suhesiak, gailu desberdinen erabiltzaile eta pasahitzen kudeaketa edo segurtasun kopia bezalakoak etortzen zaizkigu gogora. Hurrengo lerroetan gutxiago lantzen den segurtasunaren arlo batez arituko naiz. Helburua ez da baimendutako atzipenik ez ematea (beste atal batzuen ardura da hau), baizik eta sarean gertatzen ari denaren kontrola izatea, saretik igarotzen den trafikoaren kontrola izatea, alegia. Honi gehienetan NSM (*Network Security Monitoring*) izena ematen zaio. Definizio libre bat emateko, funtsean honekin lortu nahi dena da ahal den trafiko guztia biltegiratzea, aztertzea, eta trafiko susmagarriaren aurrean behar izanez gero, analisia egingo duen ingeniariari ahal den informaziorik handiena ematea.

Ikus dezagun definizioan aipatutakoa lortzeko behar dena:

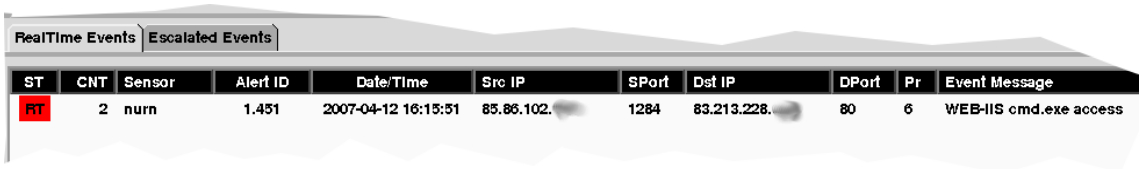
- Trafikoa biltegiratzea: Zenbat eta informazio gehiago, orduan eta analisi hobeagoa. Azpiegitura txikietan edo banda zabalera gutxi erabiltzen dutenetan erraz xamar gorde ahal izango dugu zenbait egunetan zehar sortutako trafiko guztia. Banda zabalera handiko ingurunetan hau zailagoa da, eta seguraski gordetako informazioa mugatuagoa izango da. Honi ingeleraz gehienetan "f*ull content data*" deitzen zaio. Honekin batera, "*session data*" deritzona ere gorde daiteke. Saio batean jatorri eta helburu IP helbide eta portuak, saioaren hasiera noiz izan den eta zenbat informazio igaro den ikusi daiteke normalean. Informazio hau oso lagungarria izaten da trafiko susmagarria ikusteko. Azkenik, "*statistical data*" deritzona ere erabili dezakegu. Aurreko "7.1.- Monitorizazioa" atalean esan

bezala, estatistika bidez "normaltasun egoera" zein den jakingo dugu, eta honen desbiderapenen aurrean analisia egin.

- Trafiko susmagarriaren azterketa: Trafikoa jaso eta biltegiratzen den bitartean aplikazio informatiko batek honen azterketa egin, eta susmagarria izan daitekeen edozer detektatuko du. Honela, ingeleraz "*alert data*" deritzona sortuko da. Ingeniariak, puntu honetatik aurrera, gertatutakoa aztertuko du gordetako informazioa erabiliz.

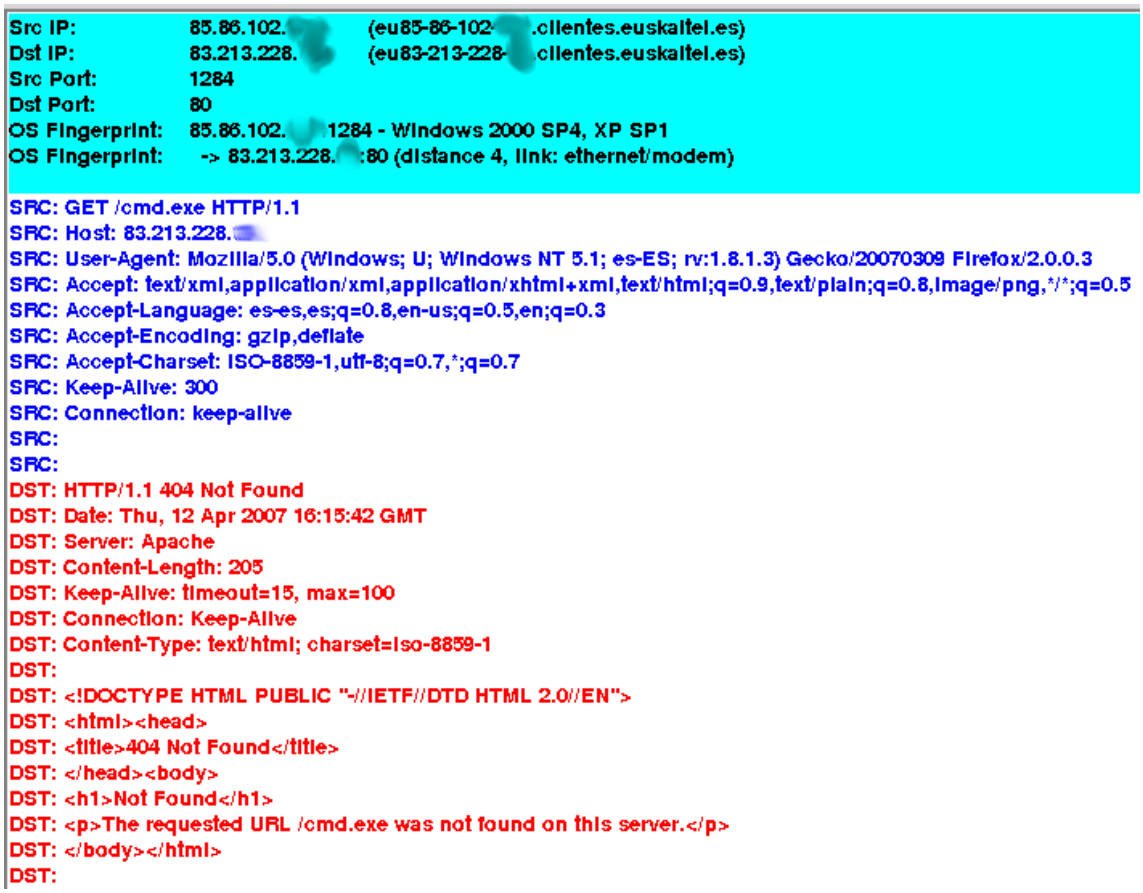
Beti bezala, software libreak tresna kopuru handia eskaintzen du NSM sistema bat egituratzeko. Interesgarrienetakoen artean *Sguil* (Sguil, 2006) dago. *Sguil*-ek *Snort*, SANCP, *Tcpflow*, *P0f* eta beste aplikazio batzuk erabiltzen ditu ingurune grafiko baten bitartez, "*alert data*", "*full content data*" edo "*session data*" moduko informazioa emateko. (Tao, 2005), (Extrusion. 2006) eta aplikazioaren web guneak erabil daitezke software honen edo NSMren inguruan informazio gehiago jasotzeko, baina hurrengo lerroetan adibide txiki bat erabiliko dut azalpen labur bat emateko.

Ondoko irudian *Sguil*-en interfaze grafikoan jasotako alerta-mezu bat ikusten da:



Irudia 32: Sguil bidez detektatutako alerta

Sguil-ek abisu bat eman du 80 portura egindako web atzipen batean. Honekin batera jatorriko IP eta portuaren informazioa ere ematen du. Izatez, *Snort* aplikazioak sortu duen mezua da hau (ikusi 4.1.6.3 atala). Azterketa sakonagoa egiteko, zein izan da web saio horretatik igaro den informazioa?

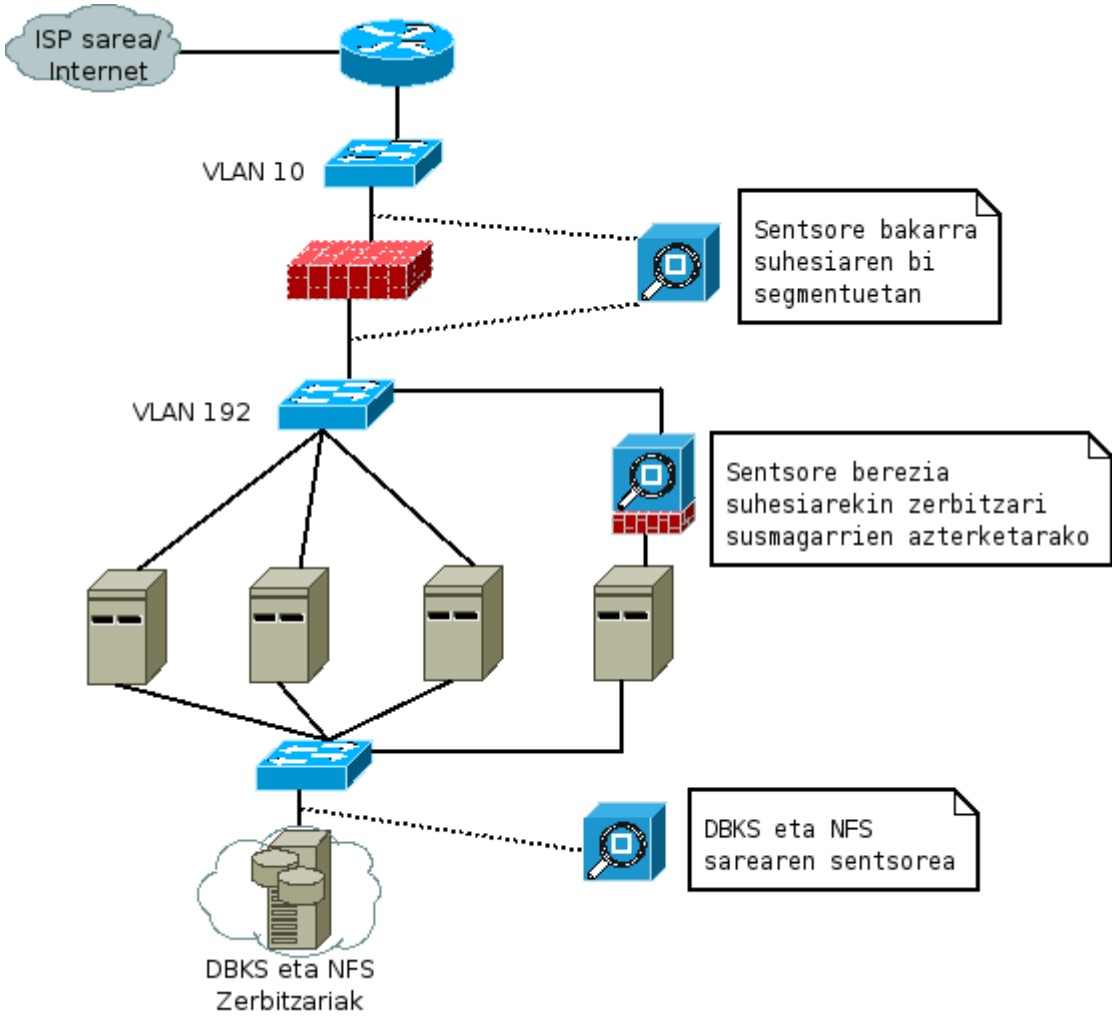


Irudia 33: Web saio bat sguil erabiliz

Tcpflow eta biltegiratutako trafikoa erabiliz, HTTP/1.1 web eskaera bat egin dela ikusten da; "*GET /cmd.exe",* hain zuzen ere. 2000/2001 urteetan oso ezagunak izan ziren kate honen bitartez *Microsoft Internet Information Server* web zerbitzariari egindako erasoak. Gure *Apache* zerbitzariak "*404 Not Found*" kodea itzuli dio nabigatzaileari, beraz, arazorik ez dela egon ondoriozta dezakegu.

Bukatu aurretik, ohar bat datuen bilketari buruz. Nondik jasotzen dira aztertuko diren datuak? Sentsorearen kontzeptua agertzen da hemen. Gure sarearen tokirik garrantzitsuenetan kokatzen diren makinak dira hauek, eta toki horietatik pasatako

trafikoa gorde eta aztertzen dutenak, *Sguil*-en kasuan emaitzak zerbitzarira bidaliz. Ahal den neurrian sarean gailu pasiboak dira, hots, matxuren aurrean ez dute sarean eraginik izango. Ikus dezagun Fnneten sarean sentsoreak kokatzeko zenbait aukera:



Irudia 34: Sentsoreak Fnneten sarean

Hasteko, ohi bezala, irudia sinplifikatu dut. Hiru sentsore mota jarri dut hemen; lehenengoa, arruntena, DBKS eta NFS makinen azpisarean dagoen trafikoa aztertzeko. Bigarrena, suhesiaren bi aldeetan jarri dut. Kontutan izan sentsoreak Linux makina

arruntak izan daitezkeela, nahi beste sare interfazekin. Hori bai, makina bakar batek, aztertu beharreko banda zabalera handia denean, agian ez du trafiko guztia aztertzeko ahalmenik izango; kasu honetan bi sentsore jartzea gomendatzen da. Normalean, sentsore mota hau sarean jartzeko *Tap* edo SPAN[88] portuak erabiltzen dira. Hirugarren sentsorea web zerbitzari baten aurrean jarri dut. Makina batean arazoren bat izatearen susmoa dagoenean sentsore/zubi/suhesi bat erabili daiteke azterketa sakonagoa egiteko. Dagoeneko badakigu zubi bat gailu gardena dela, eta kable aldaketak baino ez direla egin behar sarean, nahi dugun tokian jarri ahal izateko.

88 Tap: http://en.wikipedia.org/wiki/Network_tap

SPAN portuak Cisco gailuetan eraibiltzen dira. Beste fabrikatzaileek izen deberdinak erabiltzen dituzte kontzeptu berdintsurako.

http://www.cisco.com/en/US/products/hw/switches/ps708/products_tech_note09186a008015c612.shtml

8.- ONDORIOAK

OHARRA – Norberak jakin beharko luke zeintzuk izan diren ateratako ondorioak.

8.1.- Desbideratzearen analisia

OHARRA – Norberaren esku.

BIBLIOGRAFIA

- (Argus, 2006)

  QoSient, LLC., (2006). *Argus - Home*. http://www.qosient.com/argus/index.htm.

- (Backup, 2007)

  W. Curtis Preston (2007). *Backup & Recovery*. oreilly. ISBN: 0596102461

- (Bridge, 2006)

  linux-net.osdl.org, (2006). *Main Page - LinuxNet*.
  http://linux-net.osdl.org/index.php/Bridge.

- (Cacti, 2006)

  The Cacti Group, (2006). *Cacti: The Complete RRDTool-based Graphing Solution*.
  http://www.cacti.net.

- (Cisco, 2006)

  Cisco Systems, Inc., (2006). *Cisco Systems, Inc.*. http://www.cisco.com.

- (Ciscopress, 2007)

  Pearson Education, Cisco Press, (2007). *Cisco Press*. http://www.ciscopress.com.

- (Debian, 2006)

  Software in the Public Interest,Inc., (2006). *Debian -- The Universal Operating System*. http://www.debian.org.

- (Drbd, 2006)

  Philipp Reisner, (2006). *drbd - start*. http://www.drbd.org.

- (Extrusion. 2006)

  Richard Bejtlich (2006). *Extrusion Detection. Security Monitoring for Internal Intrusions*. Addison-Wesley. ISBN: 0321349962

- (Freebsd, 2006)

  The FreeBSD Project, (2006). *The FreeBSD Project*. http://www.freebsd.org.

- (Gfs, 2006)

  Red Hat, Inc, (2006). *redhat.com*. http://www.redhat.com/software/rha/gfs/.

- (Keepalived, 2006)

  Alexandre Cassen, (2006). *Keepalived for Linux - Linux High Availability*. http://www.keepalived.org.

- (Kernel, 2006)

  The Kernel.Org Organization, Inc., (2006). *The Linux Kernel Archives*. http://www.kernel.org.

- (Kopper, 2005)

  Karl Kopper (2005). *The Linux Enterprise Cluster*. No Starch Press. ISBN: 1593270364

- (Ldirector, 2002)

  Jacob Rief, Horms, (2002). *ldirectord*. http://www.vergenet.net/linux/ldirectord.

- (Linbit, 2006)

  LINBIT Information Technologies GmbH, (2006). *www.linbit.com: Reach your Goal with Linux - Reach Linux with LINBIT*. http://www.linbit.com/en/.

- (Linux-ha, 2005)

  Alan Robertson, (2005). *The High-Availability Linux Project*. http://www.linux-ha.org.

- (Lucas, 2004)

  Michael W. Lucas (2004). *Cisco routers for the desperate*. No Starch Press. ISBN: 1593270496

- (Lvm, 2006)

  AJ Lewis, (2006). *LVM HOWTO*. http://tldp.org/HOWTO/LVM-HOWTO/.

- (Lvs, 2006)

  Wensong Zhang, (2006). *The Linux Virtual Server Project - Linux Server Cluster for Load Balancing*. http://www.linuxvirtualserver.org.

- (Mrtg, 2006)

  Tobi Oetiker, (2006). *MRTG - Tobi Oetiker's MRTG - The Multi Router Traffic Grapher*. http://oss.oetiker.ch/mrtg/.

- (Mysql, 2006)

  MySQL AB, (2006). *MySQL AB :: The world's most popular open source database*. http://www.mysql.com.

- (Nagios, 2007)

  Ethan Galstad, (2007). *Nagios: Home*. www.nagios.org.

- (Netfilter, 2006)

  The netfilter webmaster, (2006). *netfilter/iptables project homepage - The netfilter.org project*. http://www.netfilter.org.

- (Ntop, 2006)

  ntop.org, (2006). *ntop - network top*. http://www.ntop.org.

- (Oggerino, 2001)

  Chris Oggerino (2001). *High availability network fundamentals*. Cisco Press. ISBN: 1587130173

- (Postgresql, 2006)

  PostgreSQL Global Development Group, (2006). *PostgreSQL: The world's most advanced open source database*. http://www.postgresql.org.

- (Quagga, 2006)

  maintainers@quagga.net, (2006). *Quagga Software Routing Suite*. http://www.quagga.net.

- (Reiser, 2006)

  Hans Reiser, (2006). *NAMESYS*. http://www.namesys.com.

- (Rfc, 2007)

  The Internet Society, (2007). *Search the RFC Index*. http://www.rfc-editor.org/rfcsearch.html.

- (Rfc1813, 1995)

  B. Callaghan, B., Pawlowski, P. Staubach, (1995). *NFS Version 3 Protocol Specification*. ftp://ftp.rfc-editor.org/in-notes/rfc1813.txt.

- (Rfc3530, 2003)

  The Internet Society, (2003). *Network File System (NFS) version 4 Protocol*. ftp://ftp.rfc-editor.org/in-notes/rfc3530.txt.

- (Rfc3768, 2004)

  The Internet Society, (2004). *Virtual Router Redundancy Protocol*. ftp://ftp.rfc-editor.org/in-notes/rfc3768.txt.

- (Rrdtool, 2006)

  Tobi Oetiker, (2006). *RRDtool - About RRDtool*. http://oss.oetiker.ch/rrdtool/.

- (Rsync, 2006)

  Wayne Davison, (2006). *rsync*. http://rsync.samba.org.

- (Sguil, 2006)

  Bamm Visscher, (2006). *SGUIL - The Analyst Console for Network Security Monitoring*. http://www.sguil.net.

- (Slony, 2007)

  Slony Development Group, (2007). *Slony-I*. http://slony.info.

- (Snmp, 1999)

  William Stallings (1999). *SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, third edition*. Addison-Wesley. ISBN: 0201485346

- (Snort, 2006)

  Sourcefire, Inc., (2006). *Snort - the de facto standard for intrusion detection/prevention*. http://www.snort.org.

- (Tao, 2005)

  Richard Bejtlich (2005). *The Tao of Network Security Monitoring*. Addison-Wesley. ISBN: 0321246772

- (Tcpdump, 2006)

  tcpdump.org, (2006). *TCPDUMP public repository*. http://www.tcpdump.org.

- (Tcpflow, 2003)

  Jeremy Elson, (2003). *tcpflow -- TCP Flow Recorder.* http://www.circlemud.org/~jelson/software/tcpflow/.

- (Zebra, 2003)

  IP Infusion Inc., (2003). *GNU Zebra -- routing software.* http://www.zebra.org.