TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Tietoliikennetekniikka

Antti Haakana

NAGIOS – VERKON VALVONTA

Tutkintotyö, joka on jätetty opinnäytteenä tarkastettavaksi insinöörin tutkintoa varten
Tampereella 25.2.2006

Työn valvoja: Ari Rantala
Työn ohjaaja: Riku Itäpuro, Tampereen ammattikorkeakoulu, tietokonekeskus

Tekijä: Antti Haakana
Työn nimi: Nagios – verkon valvonta

Päivämäärä: 25.2.2006
Sivumäärä: 30 sivua, 3 liitesivua
Hakusanat: Nagios, verkon valvonta, monitorointi, snmp
Koulutusohjelma: Tietotekniikka
Suuntautumisvaihtoehto: Tietoliikennetekniikka

Työn valvoja: lehtori Ari Rantala

Työn ohjaaja: Riku Itäpuro
Tampereen ammattikorkeakoulu, tietokonekeskus

Tietoliikenneverkon ja sen palveluiden toiminta on tärkeä asia niitä tuottavalle organisaatiolle. Yleensä niissä ilmenevät ongelmat näkyvät esimerkiksi palvelun toimimattomuutena tai verkon hitautena loppukäyttäjälle. Suurin osa ongelmatilanteista voidaan välttää tai jopa ennalta ehkäistä valvomalla verkon ja sen palveluiden toimintaa asianmukaisella tavalla.

Työssä käsitellään Nagios-nimistä verkonvalvontaohjelmistoa, jonka avulla voidaan valvoa myös palvelimia ja niillä tuotettuja palveluita Työn tarkoituksena on tutkia ja kehittää lisäominaisuuksia Tampereen ammattikorkeakoulun tietokonekeskuksella jo käytössä olevaan Nagios-valvontajärjestelmään.

Työn tuloksia voidaan käyttää apuna Nagios-valvontajärjestelmän muuttamiseksi entistä tehokkaammaksi ja kattavammaksi etenkin Unix- ja Linux-palvelimia ajatellen.

Author: Antti Haakana
Name of the thesis: Nagios – network monitoring

Date: 25.2.2006
Number of pages: 30 pages, 3 appendices
Keywords: Nagios, network monitoring, snmp
Degree programme: Computer Systems Engineering
Specialization: Telecommunications Engineering

Supervisor: Senior lecturer Ari Rantala

Intructor: Riku Itäpuro
Tampereen amattikorkeakoulu, computer center

Functionality of the network and its services is important matter to organization which is providing them. If these things are not functioning, the end-user sees it straight away. Most of the problems can be avoided or prevented by monitoring the state of the network and the services.

The thesis deals with Nagios network monitoring software. Nagios is also host and service monitoring program. Objective of the thesis is to research and development existing Nagios monitoring in Tampere Polytechnic's Computer Center. Results of the thesis can be used to improve Nagios monitoring especially regarding Unix- and Linux-servers.

ALKUSANAT

Tämä tutkintotyö on tehty kevään ja syksyn välisenä aikana vuonna 2005 Tampereen ammattikorkeakoulun tietokonekeskukselle.

Haluaisin kiittää työn ohjaajaa Riku Itäpuroa, jonka Linux-asiantuntemuksesta on ollut suurta apua tätä työtä tehtäessä. Kiittäisin myös Petteri Jekusta työtä koskevien kehitysehdotuksien esiin tuomisesta. Tämän työn kautta olen päässyt tutustumaan Linux-järjestelmään ja työn tekeminen on opettanut paljon uusia asioita.

Tampereella 25. helmikuuta 2006

Antti Haakana

SISÄLLYSLUETTELO

Tietotekniikka, Tietoliikennetekniikka
Antti Haakana

1 JOHDANTO

Niin suurissa kuin pienissäkin organisaatioissa lähiverkon aktiivilaitteiden, palvelinten ja palveluiden toimivuuden valvonta on tärkeää, jotta vältyttäisiin ikäviltä yllätyksiltä, jotka saattavat johtaa käyttökatkoksiin. Käyttämällä valvontaan sopivaa sovellusta, voidaan tuleva vikatilanne pystyä hyvässä tapauksessa ennakoimaan jo ennen sen ilmenemistä käyttäjätasolla. Monitorointiohjelman avulla voidaan vioista aiheutuvat käyttökatkokset saada mahdollisimman lyhyiksi, koska vian paikallistaminen helpottuu ja nopeutuu huomattavasti. Lähiverkon ja palveluiden valvontaan on kehitetty lukuisa määrä sekä kaupallisia, että ei-kaupallisia ohjelmistoja.

Tutkintotyössä keskitytään Nagios-nimiseen, vapaan lähdekoodin monitorointiohjelmaan ja tutkitaan, mitä kaikkea sillä on mahdollista valvoa. Aiheen tutkintotyöhön sain Tampereen ammattikorkeakoulun tietokonekeskuksesta, jossa haluttiin kehittää jo käytössä olevaa Nagios-valvontaa.

Työn tavoitteena oli kehittää etenkin Unix- ja Linux-palvelinten valvontaa. Palvelinten valvomiseen ei riitä vain se, että tiedetään niiden olevan käynnissä, vaan toimivuutta on hyvä tutkia palveluiden tasolla. Tehtävänäni oli myös asentaa uusi versio Nagioksesta testiympäristöön ja tutkia sen toimivuutta sekä eroja käytössä olevaan versioon. Työn tarkoituksena oli myös selvittää, miten varsinaisia tarkastuksia suorittavia lisäohjelmia voidaan kirjoittaa.

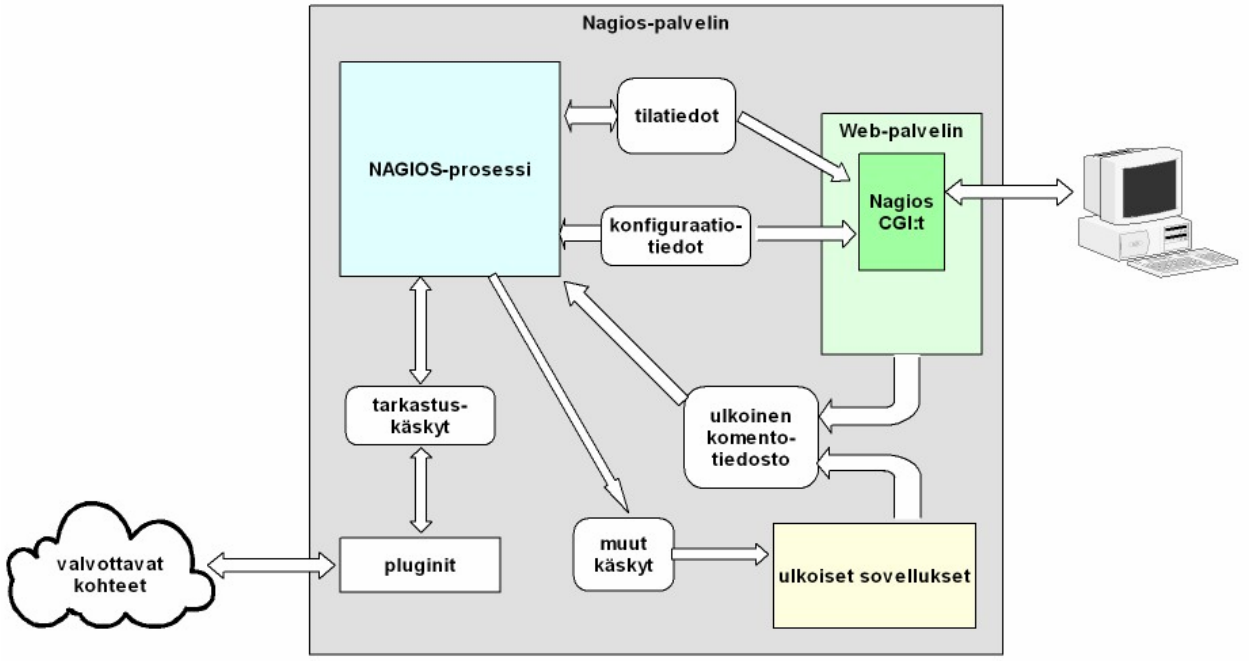
2 NAGIOS

Vuonna 1999 julkaistiin sovellus nimeltä NetSaint, jonka nimi muutettiin myöhemmin tuotemerkkisyistä Nagiokseksi. Ohjelman kehitys lähti liikkeelle tarpeesta tuottaa valvontapalveluja pienelle lähiverkolle ilman suuria kustannuksia. Nagioksen alullepanija ja pääkehittäjä on Ethan Galstad. Tähän päivään mennessä Nagios on käynyt läpi yhdeksän suurta versiomuutosta. Nagiosta käytetään kymmenissä tuhansissa yrityksissä ja organisaatioissa ympäri maailmaa ja suurimmissa raportoiduissa järjestelmissä valvotaan tuhansia kohteita. /1/

Nagios on suunniteltu toimimaan Unix- ja Linux-käyttöjärjestelmissä, ja nykyään se on sisällytetty myös useisiin suosittuihin Linux-jakeluihin, kuten esimerkiksi Debian, Gentoo ja Ubuntu /1/. Nagios on erittäin hyvin dokumentoitu projekti, jonka lisäksi tietoa ja apua saa aktiivisten sähköpostiryhmien avulla.

Nagios on vapaan lähdekoodin (Open Source) verkonvalvontasovellus, joka perustuu GPL-lisenssiin (General Public License). Tämän lisenssin alaiset ohjelmat ovat kenen tahansa levitettävissä, kopioitavissa ja muokattavissa. Lähdekoodin on avoimuus tarkoittaa sitä, että käyttäjällä pitää olla mahdollisuus saada se käyttöönsä. Näitä ohjelmia saa myös levittää maksua vastaan.

Nagioksella voidaan valvoa kaikkia lähiverkossa olevia laitteita, joilla on jonkinlainen osoite (yleensä IP-osoite), esimerkkinä verkon aktiivilaitteet, palvelimet, tulostimet ja työasemat. Valvottavat kohteet jaetaan Nagioksen kannalta palvelimiin (Host) ja palveluihin (Service). Palvelimiksi lasketaan kaikki fyysiset laitteet verkkokytkimistä työasemiin. Palvelut puolestaan ovat palvelimella tuotettuja asioita, esimerkiksi DNS-palvelut (Domain Name Service) tai jokin palvelimen tai muun laitteen ominaisuus. Jokainen palvelu liittyy johonkin palvelimeen.



Kuva 1 Nagioksen ohjelmarakenne

Nagioksen ohjelmarakenne on modulaarinen (kuva1). Nagios-taustaprosessi pitää sisällään monitorointilogiikan ja koordinoi tarkastuksia. Kaikki suoritettavat tarkastukset tehdään ulkoisten sovellusten avulla, joita ydinprosessi kutsuu. Tarkastusohjelmat eivät sisälly Nagioksen ohjelmapakettiin, vaan ovat oma erillinen osansa. Sovellukset palauttavat Nagiokselle paluuarvona tarkastuksen tuloksen ja siihen mahdollisesti liittyvät lisätiedot. Ulkoista komentotiedostoa käytetään ohjelman suorittamista koskevien käskyjen välittämiseen prosessille sekä passiivisten tarkastustulosten vastaanottamiseen.

Nagioksen keräämiä tietoja voidaan tarkastella visuaalisessa muodossa CGI:den (Common Gateway Interface) kautta Internet-selaimella. Näin toteutetun www-liittymän kautta voidaan tarkkailla esimerkiksi kaikkien palveluiden tiloja (liite1). Liitteen kuvan ensimmäisillä riveillä on Windows-koneen tarkastustuloksia, kuten prosessorikuorma sekä levyn ja muistin käyttö. Nagios tekee palvelimille määriteltyjen riippuvuussuhteiden avulla myös visuaalisen kartan palvelimista, jonka keskipisteenä on Nagios-prosessi (liite2).

Ongelmatapauksissa voidaan lähettää ilmoituksia ylläpitäjille konfiguraation mukaisesti tai suorittaa jokin komento, jonka avulla käynnistetään esimerkiksi viallinen palvelu uudelleen. Nagios on helposti integroitavissa kolmannen osapuolen sovelluksiin.

2.1 Konfigurointi

Tarkastettavista kohteista määritellään tarkastusten tekemiseksi tarvittavat tiedot konfiguraatiotiedostoihin. Nämä puolestaan esitellään erillisessä pääkonfiguraatiotiedostossa, joka sisältää myös ohjelman toiminnan ja sen käynnistymisen kannalta tärkeitä asetuksia ja valintoja. Alla on esitetty, mitä kaikkea tarkastettavalle palvelulle voidaan määritellä. Punaisella merkityt kohdat ovat pakollisia tietoja jokaiselle palvelulle. Nagioksessa voidaan käyttää mallipohjaista konfigurointia, jolloin kaikkia tietoja ei tarvitse erikseen antaa jokaiselle kohteelle, vaan niitä voidaan periä valmiiksi määritellyistä pohjista.

define service{

host_name	*“palvelimen nimi”*
service_description	*“palvelun kuvaus”*
is_volatile	*[0/1] (onko palvelu ns epävakaa)*
check_command	*“tarkastuskomento” (plugin ja sen atribuutit)*
max_check_attempts	*uudelleentarkastuksien lukumäärä ennen ensimmäisen ilmoituksen lähettämistä*
normal_check_interval	*tarkastuksien suoritusväli*
retry_check_interval	*uudelleentarkastuksien suoritusväli (jos muu kuin OK-tila)*
active_checks_enabled	*[0/1] (sallitaanko actiiviset tarkastukset)*
passive_checks_enabled	*[0/1] (sallitaanko passiiviset tarkastukset)*
check_period	*“aikajakson nimi” (tarkastusten suorittamisen aikajakso)*
parallelize_check	*[0/1] (voidaanko tark. tehdä yhtäaikaisesti muiden kanssa)*
obsess_over_service	*[0/1] (suoritetaanko tark. jälkeen pakollinen suorituskäsky)*
check_freshness	*[0/1] (tarkastetaanko palvelun tarkastuksen “tuoreus”)*
freshness_threshold	*aika, jonka kuluessa uuden passiivisen tuloksen pitäisi tulla*
event_handler	*“tapahtumankäsittelijän komento”*
event_handler_enabled	*[0/1] (sallitaanko tapahtumankäsittelijä)*
low_flap_threshold	*lepatuksen alaraja prosentteina*
high_flap_threshold	*lepatuksen yläraja prosentteina*
flap_detection_enabled	*[0/1] (sallitaanko lepatuksen tunnistus)*
process_perf_data	*[0/1] (tuotetaanko suoritusdataa)*
retain_status_information	*[0/1] (säilytetäänkö tiloihin riippuvaa tietoa uudelleenkäynnistyksen yhteydessä)*
retain_nonstatus_information	*[0/1] (säilytetäänkö tiloista riippumatonta tietoa uudelleenkäynnistyksen yhteydessä.)*
notification_interval	*aika, jonka kuluttua lähetetään uusi ilmoitus*
notification_period	*“aikajakson nimi”(ilmoitusten lähettämisen aikajakso)*
notification_options	*[w,u,c,r] (tilat, joissa ilmoituksia lähetetään)*
notifications_enabled	*[0/1] (sallitaanko ilmoitusten lähettäminen)*
contact_groups	*kontaktiryhmät*
stalking_options	*[o,w,u,c] (tilat, missä käytetään seurantaa)*

}

Palvelu voidaan määritellä epävakaaksi (volatile) sen ollessa esimerkiksi pelkästään hälytys jostain asiasta, jolloin siitä lähetetään ilmoitus välittömästi tilan vaihtuessa ”*hard-OK*”:sta (selitetään kappaleessa 2.2). Oletusasetuksena palvelut eivät ole epävakaita. Tarkastuksen jälkeisellä pakollisella suorituskäskyllä voidaan esimerkiksi lähettää jaetussa valvonnassa tarkastusten tulokset toiselle Nagios-palvelimelle. Tästä toiminnosta kerrotaan tarkemmin kappaleessa 4.

Nagios voidaan konfiguroida tunnistamaan lepatus (flap), jolla tarkoitetaan kohteen tilan vaihtumista usein. Lepatus saattaa aiheuttaa suuren määrän ilmoituksia. Lepatuksen rajat annetaan prosentteina, joihin Nagios laskee vertailuarvot 21 viimeisimmästä kohteen tarkastustuloksesta.

Ilmoituksille määritellään tilat, joissa niitä lähetetään. Valittaessa esimerkiksi ”*r*” (Recovery), lähetään ilmoitus määritellyille kontakteille palvelun palatessa toimintaan vian jälkeen. Ilmoituksille konfiguroidaan myös aika, minkä välein se lähetetään uudestaan, jos palvelu on edelleen vikatilassa. Jos ilmoitusten uudelleenlähettämisen väliksi asetetaan 0, niin ilmoitus lähetetään vain kerran. Seurannalla (stalking) tarkoitetaan seuraavassa kappaleessa esitettyjen tilatyyppien muutosten kirjaamista lokiin.

2.2 Tilatyypit

Tarkastettavien kohteiden todelliset tilat muodostuvat tilasta (esimerkiksi OK, WARNING tai CRITICAL) ja kyseisen tilan tyypistä. Tilatyypit ovat ”*hard*” ja ”*soft*”. Tilatyypit ovat oleellinen asia Nagioksen toiminnan kannalta, koska niiden perusteella lähetetään ilmoituksia ja suoritetaan tapahtumien käsittelijöitä.

Esimerkiksi kappaleen 2.1 palvelun esimerkkimäärittelyn ”*max_check_attemps*”-kohtaan määritellään uusintatarkastusten määrä siinä tapauksessa, jos tarkastuksen tuloksena on jokin muu kuin OK-tila. Uusintatarkastuksia suorittamalla voidaan välttää virheellisten ilmoitusten lähettäminen, jos tarkastuksen epäonnistuminen riippuu jostain muusta tekijästä kuin tarkastettavasta kohteesta.

Tila luokitellaan ”*Soft*”-vikatilaksi silloin, kun tarkastuksen tuloksena on saatu muu kuin OK-tila, eikä kaikkia uusintatarkastuksia ole vielä tehty. ”*Soft*”-elpymistilaksi kutsutaan puolestaan tilaa, jossa palvelin tai palvelu on palannut toimintaan vikatilasta, mutta kaikkia uusintatarkastuksia ei ole tehty. Näissä tiloissa ei lähetetä vielä ilmoituksia, mutta tapahtumankäsittelijä suoritetaan, jos sellainen on kohteelle määritelty.

Tilat muuttuvat ”*hard*”-tiloiksi vasta silloin, kun määritellyt uusintatarkastukset on suoritettu ja niistä on saatu sama tulos. ”*Hard*”-tilojen muutoksista lähetetään ilmoituksia sen mukaan, mitä kullekin kohteelle on määritelty kohdassa ”*notification_options*”.

2.3 Nagios 2.0 -version ominaisuudet

Tätä kirjoitettaessa Nagios 2.0 -versiota ei ole vielä julkaistu. Tässä kappaleessa kerrotaan muutoksista, joita uuteen versioon on tulossa. Käyttöä on testattu 2.0b3-versiolla. Konfiguraatiotiedostojen muutokset tarkastettavien kohteiden osalta on esitetty liitteessä (liite 3).

Makrot ovat läpikäyneet muutoksia, ja niitä koskevat konfiguraatiotiedot on tarkastettava ja muokattava siirryttäessä 2.0-versioon. Esimerkiksi tarkastusten tuloksena saatavien lisätietojen käyttämät makrot ovat muuttuneet palvelimilla *$HOSTINFO$*:sta $*HOSTOUTPUT$*:ksi ja palveluilla *$SERVICEINFO$*:sta *$SERVICEOUTPUT$*:ksi. Kontaktiryhmiä eli ryhmiä, joille ilmoituksia lähetetään, ei enää määritellä palvelinryhmille, vaan jokaiselle palvelimelle erikseen. Jotta palvelimia pääsee tarkastelemaan www-selaimen kautta ryhmissään, tulee käyttäjän olla oikeutettu jokaisen kyseisen ryhmän palvelimen tarkasteluun. /1/

Palvelimille voidaan konfiguroida myös ajastettuja tarkastuksia, jotka eivät olleet mahdollisia aiemmissa versioissa. Palvelimien toiminta tarkastetaan edelleenkin tarvittaessa, joten tätä optiota kannattaa käyttää harkiten, ettei kuormita turhaa Nagios-palvelinta. Lisäksi voidaan vastaanottaa myös palvelimien passiivisia tarkastustuloksia. /1/

Yleisesti ottaen enemmän tietoa säilytetään Nagioksen uudelleenkäynnistyksen yhteydessä. Esimerkkinä tarkastuksien aikataulutiedot voidaan säilyttää, jolloin tarkastuksia jatketaan siitä kohdasta, mihin ennen uudelleenkäynnistystä jäätiin. Tiedot tallennetaan uudelleenkäynnistyksessä entistä viisaammin, jolloin toiminta konfiguraatiotietojen muokkaamisen yhteydessä pitäisi yksinkertaistua.

Palveluita voidaan jakaa omiin ryhmiinsä. Palveluryhmiä voidaan tarkastella CGI:den kautta ja niihin voidaan viitata palveluiden riippuvuuksissa sekä palveluiden ilmoituksien lähettämisen porrastuksessa. Tämä ominaisuus helpottaa ja selkeyttää konfigurointia. Uutena ominaisuutena on myös niin sanottu liipaistu häiriöaika, jolla voidaan kytkeä häiriöajan alku seuraamaan toisen kohteen joustavaa häiriöaikaa. Nämä optiot ovat hyödyllisiä huoltokatkojen aikana, jolloin niitä käyttämällä vältytään turhien ilmoitusten lähettämiseltä.

Tietokantatuki (MySQL) on poistettu, mutta sopivaa tapahtuman käsittelijää käyttämällä on mahdollista edelleen käyttää tietokantoja. Tapahtumien käsittelijöille on luotu sovellusliittymä Nagios-taustaprosessin ytimeen, ja sen avulla uusien tapahtuman käsittelijöiden kehitys on entistä helpompaa.

Uutena optiona voidaan määritellä aikaikkuna, jonka kuluessa ensimmäiset tarkastukset palvelimille ja palveluille tulee suorittaa Nagioksen käynnistyttyä. Kohteille luodaan käynnistettäessä välimuistitiedosto Nagioksen toimesta. Sen pitäisi nopeuttaa CGI:den toimintaa ja sallia konfiguraatiotiedostojen editointi Nagioksen ollessa käynnissä, vaikuttamatta CGI:den ulostuloon. /1/

3 TARKASTUKSET

Tarkastukset suoritetaan erillisiä lisäohjelmia käyttämällä. Käytännössä Nagioksen avulla voidaan tarkastaa mitä vain, mille voidaan kirjoittaa oikeantyyppiset paluuarvot antava ohjelma. Tarkastusohjelmia kutsutaan englanninkielisellä nimellä *plugin*. Tarkastukset suoritetaan Nagioksen toimintalogiikassa ensisijaisesti vain palveluille. Palvelimien tarkastukset suoritetaan vasta palvelun tilan muuttuessa. Versiossa 2.0 on tosin mahdollista määritellä erikseen tarkastuksia palvelimille, mutta tämä on käytännössä turhaa, sillä jos jokin palvelu toimii palvelimella, voidaan olettaa kyseessä olevan palvelimenkin olevan toiminnassa. Toisaalta on turha valvoa sellaista palvelinta, jolla ei tuoteta merkityksellistä palvelua.

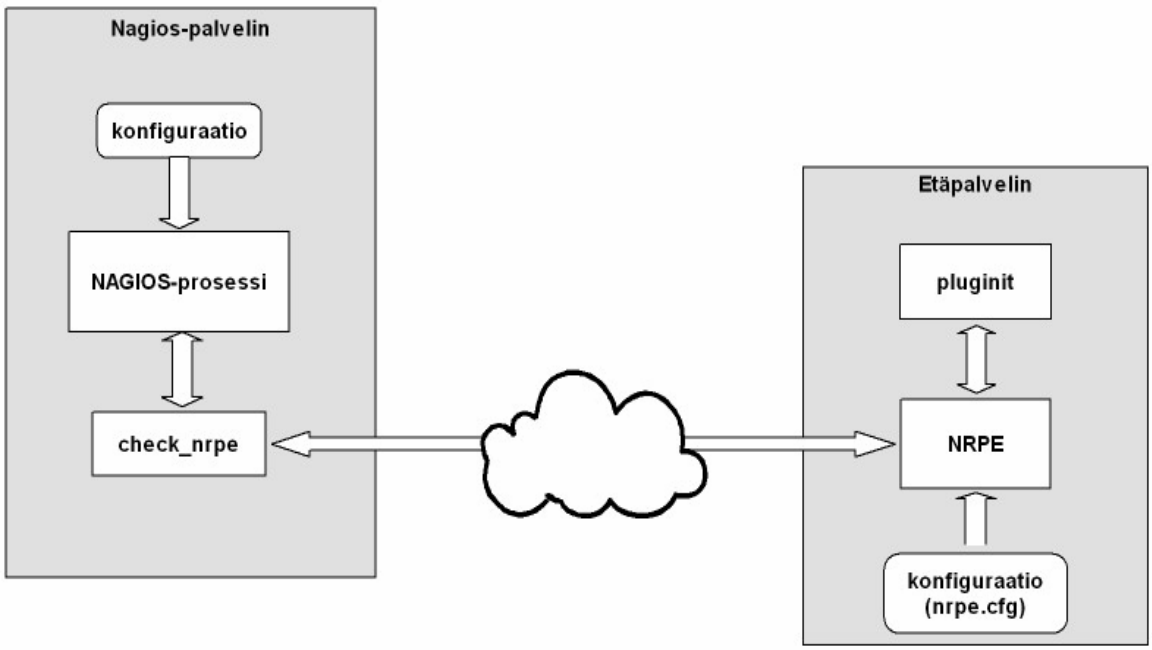
Nagios tekee kohteille suoritettaville tarkastuksille aikataulutetun järjestyksen konfiguraatiotietojen perusteella. Nagios pystyy suorittamaan samanaikaisesti useita tarkastuksia. Tarkastusten suoritustiheys voidaan konfiguroida kaikille kohteille erikseen. Esimerkkinä sähköpostin toimivuutta kannattaa valvoa tiheämmin kuin tulostimen paperimäärää.

Valmiita tarkastusohjelmia löytyy runsain määrin, esimerkiksi osoitteesta *www.nagiosexchange.org*. Ohjelmat ovat ilmaisia, kuten vapaan lähdekoodin ohjelmistoon kuuluu. Perustarkastusohjelmat sisältävä paketti on helposti ladattavissa ja asennettavissa käyttöön. Tarkastusohjelmia myös voidaan kirjoittaa itse esimerkiksi Perl-ohjelmointikielellä tai Shell-komentorivikieltä käyttäen. Jos tarkastettava kohde tai asia on yksilöllinen, se saattaa vaatia esimerkiksi hieman olemassa olevan tarkastusohjelman rakenteen muokkaamista. Tarkastusohjelmien rakennetta ja ohjelmointia on tarkasteltu vielä perusteellisemmin luvussa 3.5.

Tarkastusten konfiguraatiotiedoissa voidaan joissakin tapauksissa määritellä tulokselle rajat, joilla saavutetaan tietty tila. Esimerkiksi jäljellä olevaa levytilaa tarkastettaessa annetaan sille varoitus- ja hälytysrajat, joiden rikkoutuessa lähetetään ilmoituksia. Tarkastuksille määritellään myös, kuinka monta kertaa ne suoritetaan uudestaan, ennen kuin vikatilanteesta lähetetään ilmoituksia.

Tarkastukset voidaan jaotella aktiivisiin ja passiivisiin tarkastuksiin. Aktiivisilla tarkastuksilla tarkoitetaan tarkastuksia, jotka suoritetaan välittömästi Nagios-prosessin käskystä. Passiivinen tarkastus suoritetaan paikallisesti etäkoneella ja sen tulos lähetetään Nagios-prosessille verkon yli. Tällaisessa tapauksessa Nagios-palvelin konfiguroidaan vastaanottamaan passiivisten tarkastusten tuloksia, ja käytännössä se lukee tulokset ulkoisesta komentotiedostosta.

3.1 NRPE

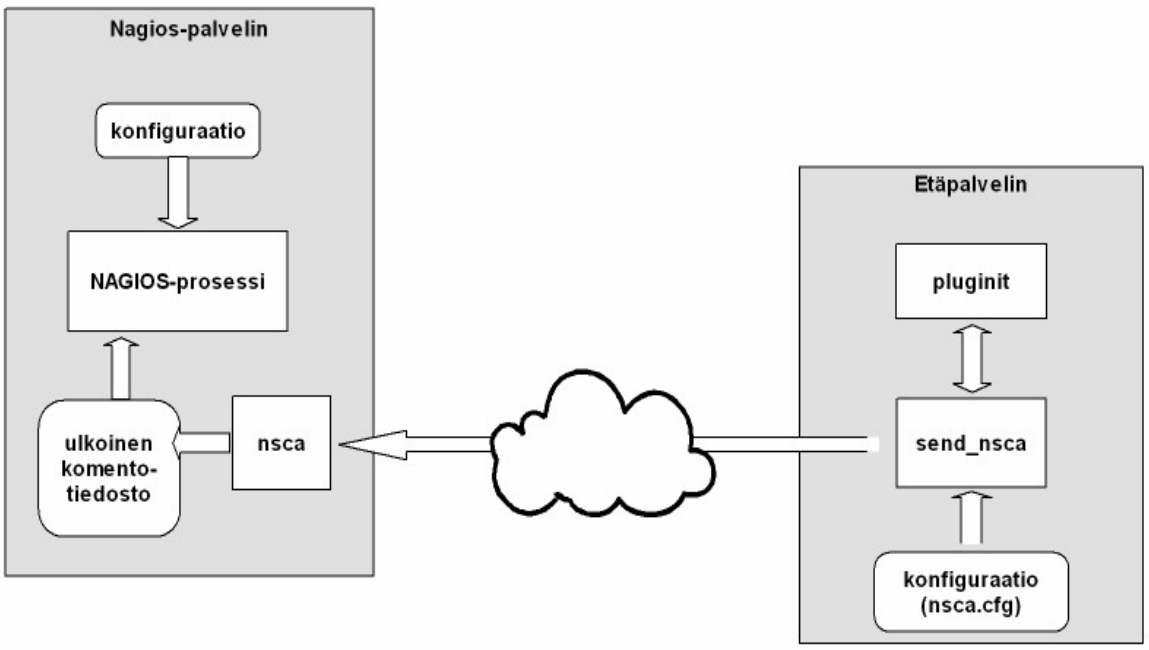


Kuva 2 Tarkastusten suorittaminen NRPE:n avulla

Esimerkiksi prosessorikuorman tai levytilan tarkastamiseksi täytyy tarkastusohjelma suorittaa paikallisesti tarkastettavalla palvelimella. Lisäosan nimi NRPE on lyhenne, joka tulee sanoista Nagios Remote Plugin Executor. Lisäosista käytetään yleisesti englanninkielistä nimeä ”*add-on*”. NRPE on aktiivinen lisäosa, joka mahdollistaa tarkastusohjelmien suorittamisen etäpalvelimella sijaitsevista kohteista (kuva 2). Nagios-palvelin antaa tarkastuksen suorituskäskyn ”*check_nrpe*”-nimiselle lisäohjelmalle, joka lähettää tiedot verkon yli etäpalvelimelle asennetulle NRPE-prosessille. Tämä puolestaan suorittaa etäpalvelimella tarkastusohjelman. Tarkastuksen tulos lähetetään samaa ketjua pitkin takaisin Nagiokselle.

NRPE:n tietoturvaa voidaan parantaa käyttämällä niin sanottua IP-pääsylistaa (TCP-wrapper). Tällöin etäkoneelle voidaan määritellä sovelluskohtaisesti NRPE:lle yhteys sallittuihin osoitteisiin (hosts.allow-listalla) ja kieltää yhteydet (hosts.deny-listalla) muista osoitteesta /1/. NRPE:n verkkoliikenne on myös mahdollista konfiguroida käyttämään SSL-salausta.

3.2 NSCA



Kuva 3 Tarkastusten suorittaminen NSCA:n avulla

Lisäosan nimi tulee sanoista Nagios Service Check Acceptor. NSCA:ta käytettäessä etäpalvelimelle asennetaan ”*send_nsca*”-niminen passiivinen tarkastusprosessi. Prosessi suorittaa siellä tarkastuksia lisäohjelmien avulla ja lähettää niiden tulokset Nagios-palvelimelle. Tulokset otetaan vastaan ”*nsca*”-lisäosan avulla (kuva 3), joka kirjoittaa ne ulkoiseen komentotiedostoon. Nagios-prosessi konfiguroidaan lukemaan tätä komentotiedostoa tietyin väliajoin. NSCA:n hyvä puoli on, että se kuluttaa Nagios-palvelimen resursseja huomattavasti vähemmän kuin aktiiviset tarkastukset, koska se vain vastaanottaa tarkastusten tuloksia. Tämä pätee myös muilla tavoin suoritettuihin passiivisiin tarkastuksiin.

Tietoturvan lisäämiseksi on syytä huomioida muutamia asioita NSCA:ta käytettäessä. Jotta voitaisiin välttää virheellisten tietojen vastaanottaminen, ja pahimmassa tapauksessa niiden takia suoritetut toimenpiteet (esimerkiksi tapahtumankäsittelijän käynnistyminen), voidaan NSCA:n tiedonsiirto asettaa käyttämään salausta. Käytännössä tämä tarkoittaa sitä, että tarkastuksen tulos pakataan etäpalvelimella pakettiin siitä lasketun CRC-32-tarkistussumman kanssa.

Tämän jälkeen lähetettävä paketti salataan käyttämällä konfiguraatiotiedostoon määriteltyä salasanaa. Salausalgoritmi voidaan valita 26:sta eri vaihtoehdosta, esimerkkeinä seuraavat algoritmit: DES, 3DES, CAST, xTEA, Twofish, LOKI97, RJINDAEL, SERPENT, GOST, SAFER/SAFER+. Vastaanottopäässä salattu paketti puretaan käyttämällä samaa algoritmia ja salasanaa. Lisäksi CRC-32-tarkistussumma lasketaan uudestaan ja sitä verrataan lähetyspäässä laskettuun. Jos tarkistussumma on sama, niin vastaanotetun paketti hyväksytään ja todetaan sen olevan oikeasta lähteestä. /3/

3.2 Windows-palvelinten tarkastukset

Windows-palvelimilla tarkastusten suorittamiseksi voidaan käyttää esimerkiksi NSClient- tai NC_Net-lisäohjelmaa. NSClient on kehitetty aktiivisten tarkastusten tekoon, kun taas NC_Netin avulla pystyy suorittamaan myös passiivisia tarkastuksia. Nagios-prosessi käyttää NSClientin kanssa kommunikointiin ”*check_nt*”-lisäohjelmaa. Palvelinten välinen yhteys voidaan suojata salasanalla.

Tietoturvan kannalta ajateltuna passiivisen NC_Netin käyttö on turvallisempaa kuin NSClientin. Tiedonsiirto voidaan salata käyttämällä NC_Netin kanssa ulkoista ”*send_NSCA*”-lisäohjelmaa ja NSCA:n Win32Clientia tietojen salaamiseksi samalla periaatteella kuten edellisessä kappaleessa on esitetty.

3.4 SNMP

SNMP (Simple Network Management Protocol) on TCP/IP-verkkojen hallinnassa käytettävä tietoliikenneprotokolla. SNMP:tä apuna käyttämällä on myös mahdollista suorittaa tarkastuksia Nagioksen kanssa.

SNMP:lla on puumainen rakenne MIB (Management Information Base), jossa jokaisella kohteella on identifioiva tieto. Tätä tietoa kutsutaan myös OID:ksi (Object Identifier). OID on numeerinen arvo MIB:stä. SNMP:stä käytetään valvonnan yhteydessä luku- ja keskeytysominaisuutta. Keskeytysominaisuutta (*trap*) voidaan käyttää siten, että asennetaan palvelimelle SNMP-keskeytysohjelma, joka konfiguroidaan lähettämään vikatilanteessa hälytys Nagiokselle.

Lukuominaisuuden avulla on esimerkiksi verkon aktiivilaitteista ja tulostimista saatavilla paljon tietoa. Verkkokytkimestä voidaan tutkia esimerkiksi verkkoliikenteen määrää, yksittäisten porttien tiloja, sekä virtalähteiden tuulettimien toimivuutta. SNMP-tarkastuksien suorittamista varten on olemassa valmis lisäohjelma Nagiokselle. Alla on esitetty esimerkki, millaisella komennolla HP:n Procurve-kytkimestä saadaan yksittäisen portin tila selville käyttämällä Nagioksen ”*check_snmp*”-lisäohjelmaa. Lisäohjelmia voi myös käyttää testaamisessa suoraan komentoriviltä, kuten esimerkkikin on esitetty.

./check_snmp -H xxx.xxx.xxx.xxx -C community -o .1.3.6.1.2.1.2.2.1.8.portin_numero

Jossa,

- *H on tarkastettavan laitteen IP-osoite*
- *C on Community name eli eräänlainen salasana, jotta tietoja päästään lukemaan*
- *o on OID, eli laitteen portille asetettu yksilöivä tunnus*

SNMP:n käyttö on mahdollista myös palvelimilla sijaitsevien kohteiden tarkastamiseen. Tällöin palvelimelle asennetaan SNMP-agentti, joka toimii rajapintana verkon ja suoritettavien tarkastusten välillä. SNMP:ltä kysytään tiettyyn OID:hen liittyvää tulosta agentilta, jonka se palauttaa. Tässä tapauksessa tarkastusten tulokset voidaan lukea etäpalvelimelta, esimerkiksi väliaikaistiedostosta, tai suorittaa jokin tarkastustoimenpide annetun OID:n perusteella.

3.5 Tarkastusohjelmien kirjoittaminen

Kriittisin osa tarkastusohjelman toiminnan takaamiseksi Nagioksen kanssa, ovat ulostuloarvot, jotka Nagiokselle palautetaan. Ohjelman olisi hyvä antaa muitakin paluuarvoja kuin esimerkiksi ”OK”, jos mahdollista. Nagios lukee yhden rivin tulostuskanavasta STDOUT, johon pitäisi saada kaikki tarpeellinen tieto mahtumaan. Ulostulon tulisi olla muotoa ”*tila:lisäinformaatio*”. Paluukoodit ovat POSIX-spesifikaation mukaisesti positiivisia (taulukko 1). Käytännössä paluukoodi annetaan niin sanottuna exit-tilana, esimerkiksi poistutaan Shell-ohjelmasta komennolla *exit 0;*.

Taulukko 1 Tarkastusohjelmien paluukoodit /2/

Paluukoodi	Tila	Tilan kuvaus
0	*OK*	Tarkastus suoritettu ja kohde toiminnassa
1	*Warning*	Tarkastus suoritettu, mutta tulos varoitusrajalla tai kohde ei toiminut kunnolla
2	*Critical*	Havaittu tuloksen olevan hälytysrajalla tai kohde ei toiminut
3	*Unknown*	Tarkastusohjelmalle on annettu vääriä parametreja tai se ei ole pystynyt tarkastamaan tilaa

Vaikka tarkastusohjelmien ohjelmointikieltä ei ole juurikaan rajoitettu, niin Perl-kielen käyttö on suositeltavaa ainakin pidempiä ohjelmia kirjoitettaessa. Perlillä saadaan aikaan suoritusajaltaan nopeita ohjelmia. Tämän lisäksi Nagios tarjoaa sisäänrakennettua Perl-tukea, joka lisää osaltaan tarkastusohjelmien suoritusnopeutta. Perl-kielisiin tarkastusohjelmiin tulee ohjelmakoodissa ladata utils.pm-moduuli. Moduuli tulee perustarkastusohjelmapaketin mukana.

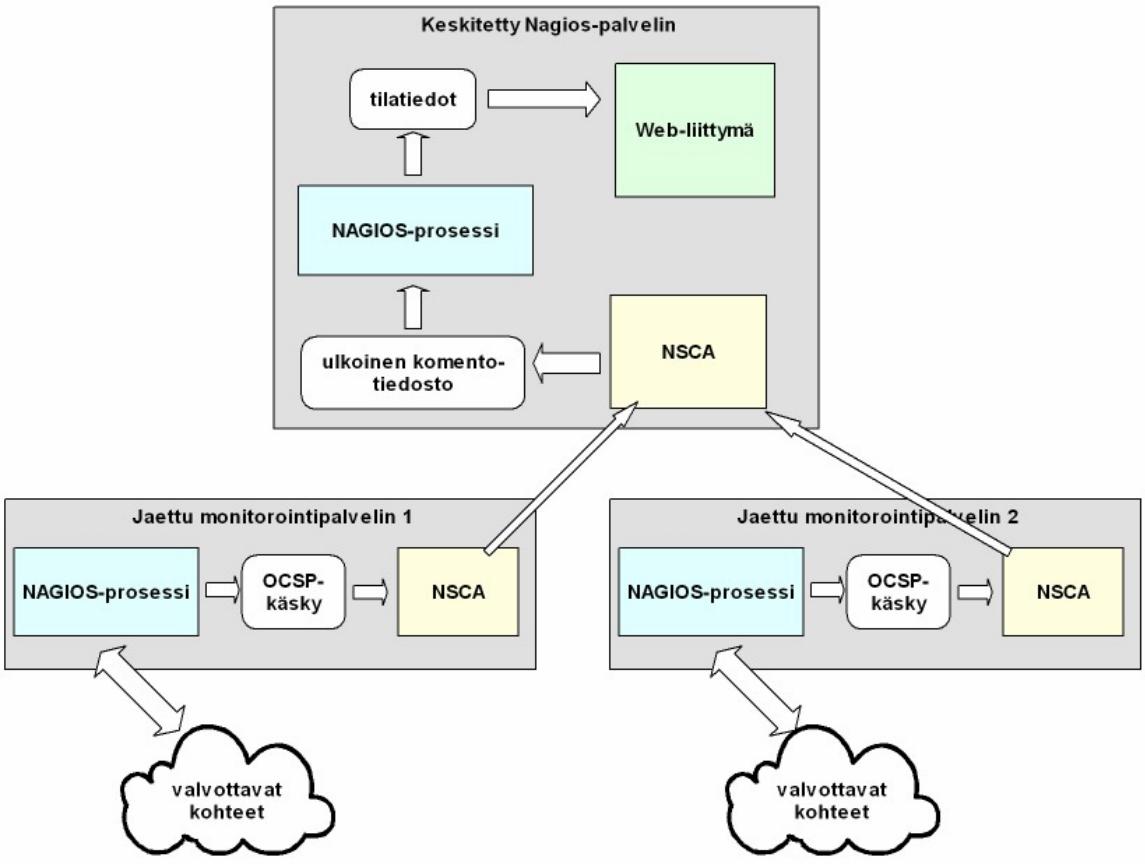
Perl ohjelmasta paluuarvo tulee antaa seuraavassa muodossa: ”*exit $ERRORS{'tila'}*”. Seuraavalla sivulla ovat esitettynä tarkastusohjelman tärkeimmät koodirivit. Koodia voidaan käyttää peruspohjana uusia Perl-kielisiä tarkastusohjelmia tehtäessä.

```
use strict;
use lib utils.pm ;
use utils qw(%ERRORS &print_revision &support);

…varsinainen ohjelmakoodi haluttuja tarkastuksia varten…

#paluuarvot
if (ehtolauseke1) {
        print “$tila:lisatiedot1”;
        exit $ERRORS{'OK'};
}
elseif (ehtolauseke2) {
        print “$tila:lisatiedot2”;
        exit $ERRORS{'WARNING'};
}
elseif (ehtolauseke3) {
        print “$tila:lisatiedot3”;
        exit $ERRORS{'CRITICAL'};
}
elseif (ehtolauseke4) {
        print “$tila:lisatiedot4”;
        exit $ERRORS{'UNKNOWN'};
}
```

4 NAGIOKSEN LISÄOMINAISUUKSIA

Tässä kappaleessa perehdytään Nagioksen ominaisuuksiin, jotka ovat tukena järjestelmän jatkokehitystä ajatellen. Pääasiassa käsitellään useammalla Nagios-palvelimella suoritettavaa valvontaa sekä ilmoituksien lähetettämisen lisäominaisuuksia. Varsinaiset kehitysehdotukset on koottu kappaleeseen 5.

4.1 Jaettu valvonta



Kuva 4 Jaetun valvonnan periaatekuva

Kun valvottavia kohteita on runsaasti, voidaan tästä syntyvää kuormaa jakaa usean valvontapalvelimen kesken. Jaetun valvonnan toteutuksessa käytetään yhtä keskitettyä Nagios-palvelinta, joka kerää yhteen kahden tai useamman jaetun Nagios-prosessin tulokset (kuva 4).

Jokaisen tarkastuksen jälkeen jaetulla monitorointipalvelimilla suoritetaan OCSP-käsky (Obsessive Compulsive Service Processor). Tämä välittää tulokset NSCA:lle, joka puolestaan huolehtii niiden siirrosta keskitetylle Nagios-palvelimelle.

Jaetut valvontapalvelimet hoitavat vain tarkastukset ja niistä saatujen tuloksien lähettämisen eteenpäin. Niiden ei tarvitse lähettää ilmoituksia eikä näyttää tiloja www-liittymän kautta. Jaetuille palvelimille konfiguroidaan vain kohteet joiden tarkastukset kukin palvelin hoitaa. Keskitetyllä palvelimella puolestaan tulee olla konfiguroituna myös kaikki jaettujen palvelinten tarkastuskohteet, jotta se voi vastaanottaa näistä passiivisia tarkastustuloksia.

Passiivisia tarkastuksia käytettäessä saattaa jaetun monitorointipalvelimen vikatilanteessa käydä niin, että tarkastustulokset lakkaavat saapumasta keskitetylle monitorointipalvelimelle. Tämän tilanteen havaitsemiseksi on syytä käyttää keskitetyllä palvelimella tarkastustulosten tuoreuden valvontaa. Se asetetaan päälle sallimalla palveluille ”*check_freshness*”-optio. Viimeisimmän tarkastustuloksen ikää verrataan kohdassa ”*freshness_threshold*” määritettyyn arvoon ja jos ikä ylittää tämän, niin tuloksen katsotaan olevan vanhentunut.

Tässä tapauksessa suoritetaan keskitetyltä monitorointipalvelimelta ”*check_command*”-kohdassa määritetty tarkastuskäsky. Tarkastus tehdään siitä huolimatta, vaikka aktiiviset tarkastukset olisivatkin poissa käytössä. Tarkastuskäskyä käyttämällä voidaan näin suorittaa kyseisen palvelun tarkastus suoraan keskitetyltä palvelimelta. Jos se ei ole mahdollista, niin vaihtoehtona on luoda tällaisissa tapauksissa käytettäväksi tarkastusohjelma, joka palauttaa esimerkiksi seuraavanlaisen tuloksen: ”*CRITICAL: Palvelun tarkastustulokset ovat vanhentuneet.*” Näin saadaan ainakin ilmoitus vikatilanteesta aikaiseksi.

4.2 Redundantti valvonta

Redundantin monitorointipalvelimen liittäminen Nagios-valvontajärjestelmään lisää sen vikasietoisuutta. Esimerkiksi valvontapalvelimen rikkoontuessa, tai verkkoyhteys siihen menettäessä, voidaan valvontaa jatkaa toisella palvelimella. Kun käytössä on kaksi

palvelinta, voidaan niitä kutsua termeillä ”isäntä” ja ”orja”. Molemmat palvelimet suorittavat samanaikaisesti tarkastuksia. Isäntäpalvelimen toimiessa orjapalvelimen ilmoitusten lähettäminen on poistettu käytöstä, ja sen ulkoisten komentojen vastaanottaminen on sallittu.

Tarkastettavien kohteiden lisäksi orjapalvelin tarkastaa myös isäntäpalvelimen ja sen Nagios-prosessin tilat. Näille tarkastuksille on määritelty tapahtuman käsittelijät. Tapahtumien käsittelijöille annetaan parametreina tilatieto sekä tilatiedon tyyppi. Periaatteessa isäntäpalvelimella ei tarvitse olla minkäänlaista tietoa orjapalvelimen olemassaolosta. Seuraavaksi on esitetty orjapalvelimella tarvittavien tapahtumankäsittelijöiden ohjelmalliset rakenteet, jotka ovat toteutettu Shell-skripteinä.

Isäntäpalvelimen tarkastuksen tapahtumankäsittelijät /1/:

```
#!/bin/sh
# Suoritetetaan vain HARD-tiloissa
case "$2" in
HARD)
        case "$1" in
        DOWN)
                #Isäntäpalvelin on alhaalla, joten orjapalvelimesta täytyy tehdä isäntäpalvelin.
                #Käytännössä tämä hoidetaan sallimalla ilmoitusten lähettäminen.
                / asennushakemisto/eventhandlers/enable_notifications
                ;;
        UP)
                # Isäntäpalvelin on jälleen toiminnassa.
                #Poistetaan ilmoitusten lähettäminen käytöstä.
                /asennushakemisto/eventhandlers/disable_notifications
                ;;
        esac
        ;;
esac
exit 0
```

Nagios-prosessin tarkastuksen tapahtumankäsittelijä /1/:

```
#!/bin/sh
# Suoritetetaan vain HARD-tiloissa
case "$2" in
HARD)
        case "$1" in
        CRITICAL)
                # Isäntäpalvelimen Nagios-prosessi ei ole ylhäällä.
                # Sallitaan ilmoitusten lähettäminen
                / asennushakemisto/eventhandlers/enable_notifications
                ;;
        WARNING)
        UNKNOWN)
                # Ei tiedetä, onko prosessi ylhäällä.
                #Tähän voi päättää, ryhdytäänkö toimenpiteisiin.
                ;;
        OK)
                # Isäntäpalvelin on jälleen toiminnassa.
                #Poistetaan ilmoitusten lähettäminen käytöstä.
                / asennushakemist/eventhandlers/disable_notifications
                ;;
        esac
        ;;
esac
exit 0
```

Näillä tapahtumankäsittelijöillä varmistetaan, että orjapalvelin ottaa vastuun ilmoitusten lähettämisestä, jos isäntäpalvelin tai sen Nagios-prosessi ei ole käynnissä. Ne toimivat myös isäntäpalvelimen palatessa toimintaan poistamalla ilmoitusten lähetyksen käytöstä.

Tällä tavoin toteutettu redundanssin lisääminen ei ole kuitenkaan täysin ongelmaton. Kun isäntäpalvelimen toiminta loppuu, niin orjapalvelimen toiminta ei ala välittömästi, vaan siirrokseen kuluu aina jonkin verran aikaa. Tätä aikaa voidaan pienentää tihentämällä isäntäpalvelimelle tehtävien tarkastuksien väliä ja pienentämällä uusintatarkastuksien määrää, jolloin tilan tyyppi muuttuu nopeammin ”hard”:ksi. Myös ulkoisen komentotiedoston tarkastamisen tihentäminen pienentää siirroksesta aiheutuvaa aikaa. /1/

4.3 Valvonnan siirto vikatilanteessa

Redundantin valvonnan huonona puolena on myös verkkoliikenteen ja palvelinkuorman kasvu, koska kaksi palvelinta suorittaa tarkastuksia samanaikaisesti kaikille kohteille. Valvonnan siirto vikatilanteessa on samankaltainen operaatio edellä kuvatun redundantin monitoroinnin kanssa.

Periaatteellisena erona on se, että orjapalvelin ei suorita tarkastuksia, vaan on niin sanotusti odotustilassa. Orjapalvelimen on tarkoitus aloittaa tarkastukset vasta isäntäpalvelimen joutuessa vikatilaan.

Valvonnan siirron orjapalvelimelta poistetaan käytöstä tarkastusten suorittaminen ja ilmoitusten lähettäminen. Se konfiguroidaan tutkimaan vain ulkoista komentotiedostoa. Orjapalvelimella suoritetaan *cron:ssa* ohjelmaa, joka tutkii isäntäpalvelimen Nagios-prosessin tilaa NRPE-lisäohjelman avulla. Jos tämä ohjelma havaitsee, että Nagios ei ole käynnissä isäntäkoneella, ohjelma aktivoi orjapalvelimen asettamalla sen tarkastusten suorituksen ja ilmoitusten lähettämisen toimintaan. Kun isäntäpalvelin palautuu vikatilasta, toimitaan päinvastoin ja palautetaan palvelinten alkuperäinen toimintatapa.

Tällä tavoin valvonta siirrettäessä ongelmaksi muodostuu se, että orjapalvelimella ei ole aloittaessaan minkäänlaisia tilatietoja. Tämän voi ratkaista esimerkiksi sallimalla isäntäpalvelimelta kaikki tarkastusten tulosten lähettämisen OCPS-käskyn avulla orjapalvelimelle käyttäen NSCA-lisäosaa. Näin saadaan orjapalvelimelle tilatiedot talteen, jolloin sen ei tarvitse aloittaa valvontaa aivan nollatilasta vikatilanteen ilmetessä.

4.4 Ilmoituksien porrastaminen

Palvelimien ja palveluiden vikatilanteista lähetettäville ilmoituksille määritellään normaalisti ajanjakso jona ilmoituksia lähetetään, montako kertaa kohde tarkastetaan ennen kuin ensimmäinen ilmoitus lähetetään ja ilmoitusten lähetystiheys. Lisäksi ilmoituksille voidaan erikseen määritellä järjestysnumerosta riippuva porrastus, jolloin ilmoitukset lähetetään eri kontaktiryhmille. Porrastuksessa voidaan myös muuttaa ilmoitusten lähetystiheyttä. Seuraavaksi on esitetty esimerkki ilmoitusten porrastuksen konfiguroinnista.

```
define serviceescalation{
        host_name                www_palvelin
        service_description      HTTP
        first_notification  3
        last_notification   5
        notification_interval    90
        contact_groups           nt-yllapito,nt-pomo
        }

define serviceescalation{
        host_name                www_palvelin
        service_description      HTTP
        first_notification  6
        last_notification   10
        notification_interval    60
        contact_groups           nt-yllapito,nt-pomo,kaikki
        }
```

Esimerkissä kaksi ensimmäistä ja kymmenen jälkeen tulevat ilmoitukset käsitellään palvelun normaalin konfiguraation mukaisesti. Ilmoitukset kolmesta viiteen lähetetään 90 minuutin välein. Ilmoitukset kuudesta kymmeneen lähetetään 60 minuutin välein ja lisäksi nämä lähetetään ryhmälle ”kaikki”. Jos ”*last_notification*”-kohtaan määritellään 0, niin siitä porrastuksesta lähetetään vain yksi, viimeinen ilmoitus. Jos esimerkiksi edellisten lisäksi määriteltäisiin samalle palvelulle vielä yksi porrastus, joka alkaisi 11 ilmoituksesta ja päättyisi nollaan, tuottaisi tämä viimeisen ilmoituksen kohteesta.

Kohteiden elpymistapauksissa porrastus toimii siten, että ilmoitus OK-tilasta lähetetään vain niille ryhmille joille on lähetetty ilmoitus CRITICAL-tilasta, vaikka jälkimmäinen osuisikin järjestysnumeroltaan jo seuraavaan porrastuksen määritelmään.

4.5 Riippuvuussuhteet

Palveluille on myös mahdollista asettaa riippuvuussuhteita, joissa yhden palvelun vikatilanne vaikuttaa toiseen. Näin voidaan pienentää ilmoitusten ja aktiivisten tarkastuksien määrää. Riippuvuussuhteiden voimassaollessa ei riippuvaista palvelua tarkasteta, jos tiedetään sen palvelun mistä ollaan riippuvaisia, olevan vikatilassa. Esimerkkinä on esitetty palvelun riippuvuuden määrittely.

```
define servicedependency{
        dependent_host_name             palvelimen_nimi
        dependent_service_description   palvelun_nimi
        host_name                       palvelimen_nimi
        service_description             palvelun_nimi
        execution_failure_criteria      [o,w,u,c,n]
        notification_failure_criteria   [o,w,u,c,n]
}
```

”Dependent”-sanoilla merkittyihin kohtiin määritellään riippuvainen palvelu ja palvelin, jolla se sijaitsee. Kahteen seuraavaan kohtaan tulee sen palvelun tiedot, josta ollaan riippuvaisia. Lisäksi voidaan määritellä kriteerit, missä tiloissa ollaan riippuvaisia ja missä lähetetään ilmoituksia. Tilojen määritelmät ovan seuraavanlaiset:

o → OK-tila

w → WARNING-tila

u → UNKNOWN-tila

c → CRITICAL-tila

n → suoritetaan aina

5 NAGIOS-VALVONNAN KEHITTÄMINEN TAMPEREEN AMMATTIKORKEAKOULUSSA

Tähän kappaleeseen on kerätty kehitysehdotuksia Tampereen ammattikorkeakoulun tietokonekeskuksen Nagios-valvontajärjestelmää koskien. Palveluista erityisesti www-palveluiden, esimerkiksi sähköpostin ja intran, tarkastaminen helpottaisi ongelmien havaitsemista ja tätä kautta palvelutason paranemista.

5.1 Verkon ja sen aktiivilaitteiden valvonta

Nagiosta pystyisi hyödyntämään verkkopuolella tekemällä kytkinten porteista erikseen tarkastettavia kohteita (palvelimia). Kaikkien porttien siirto valvontaan ja tilanteen pitäminen ajantaseina saattaisi olla liian työlästä, mutta tätä kannattaisi miettiä tärkeiden ja vähän muutoksia omaavien kytkentöjen suhteen. Näin saataisiin palvelimet yhdistettyä kytkimien tiettyihin portteihin, joka puolestaan auttaisi vian ratkaisemisessa. Vika voidaan näin paikallistaa Nagioksen avulla pienempään joukkoon ja esimerkiksi viallinen tai irronnut verkkokaapeli havaitaan nopeasti. Myös Nagioksen tekemästä visuaalisesta kartasta saisi tällä tavoin hyvin yksityiskohtaisen.

5.2 Web-sivun sisällön tarkastus

Nagioksen valvontaan Tampereen ammattikorkeakoulussa voisi lisätä myös tärkeitä palveluita, esimerkiksi sähköpostin sisäänkirjautumissivun. Sähköpostijärjestelmä saattaa joutua tilaan, jossa postit toimivat erillisillä postiohjelmilla hyvin, mutta webmailin sisäänkirjautumissivu on poissa käytöstä. Sivun olemassaolon voi tarkastaa käyttämällä ”*check_http*”–tarkastusohjelmaa. Tarkastus konfiguroidaan ”*checkcommands.cfg*”–tiedostoon seuraavasti:

```
# https sivun tarkastuksen maarittely
define command{
        command_name    check_https2
        command_line    /asennushakemisto/check_http --ssl -H $HOSTADDRESS$ -I
                        $HOSTADDRESS$ -s $ARG1$
}
```

Konfiguroinnissa määritetään kutsuttavan tarkastuksen nimi, ja mitä tarkastusohjelmaa käytetään (*check_http*). Lisäksi tarkastusohjelmalle annetaan parametreina tieto sivun tiedonsiirrossa käytettävästä salauksesta (SSL), tarkastettavan palvelimen IP-osoitteesta ja merkkijonosta, jota sivulta haetaan. Seuraavaksi on esitetty varsinainen tarkastuksen määrittely konfiguraatiotiedostoon *services.cfg*:

```
#webmailin kirjautumissivun  toiminnan tarkastus
define service {
      use                     palvelu-pohja
      host_name               postipalvelin
      service_description     webmail-login-sivu
      check_command           check_https2!TAMK
 }
```

Tarkastuksen konfiguroinnissa ensimmäisenä on määritelty käytettävä mallipohja, jotta kaikkia kohtia ei tarvitse erikseen määritellä jokaiselle kohteelle. Seuraavaksi kerrotaan palvelimen nimi, jolla kirjautumissivu sijaitsee, ja annetaan tarkastettavalla palvelulle identifioiva kuvaus. Lisäksi määritetään tarkastuskomennon argumenttina sivulta etsittävä merkkijono ”TAMK”. Argumenttien erotukseen käytetään huutomerkkiä. Jos tarkastus ei löydä sivulta merkkijonoa, päätetään sen olevan *”CRITICAL”*-tilassa, ja voidaan lähettää asiasta ilmoitus.

5.3 Web-sivun SSL-sertifikaatin tarkastus

SSL-protokolla on nykyisin yksi tavallisimpia tapoja suojata tietoliikennettä IP-verkoissa. SSL-sertifikaattia tarvitaan, kun halutaan salata palvelimen ja selaimen välinen liikenne (https). SSL-sertifikaatti sisältää salausavaimen, tiedon palvelimesta jolle se on myönnetty, sekä myöntäjän allekirjoituksen.

Web-sivun SSL-sertifikaatin voimassaolo voidaan tarkastaa Nagioksen avulla käyttämällä samaa tarkastusohjelmaa kuin sivun sisällön tarkastamiseksi. Seuraavaksi on esitetty määrittelyt sertifikaatin tarkastamiseksi postipalvelimelta.

checkcommands.cfg:

```
# ssl-sertifikaatin tarkastuksen maarittely
define command{
        command_name        check_cert
        command_line        / asennushakemisto/check_http --ssl -H $HOSTADDRESS$ -I
                            $HOSTADDRESS$ -C $ARG1$
}
```

services.cfg:

```
#webmailin sertifikaatin tarkastus
define service {
        use                 generic-service
        host_name           postipalvelinl
        service_description webmail-sertifikaatti
        check_command       check_cert!10
}
```

Tarkastukselle annetaan parametrina aika päivinä, jonka sertifikaatin tulee olla voimassa. Kun tämä luku alittuu (esimerkissä 10), niin seuraa *”CRITICAL”* –tila ja ilmoituksien lähettäminen määrittelyn mukaan. Tarkastusohjelma antaa myös paluuarvona sertifikaatin voimassaolopäivien lukumäärän, joka näkyy Nagioksen web-liittymän kautta.

5.4 UNIX-palvelimien valvonta

Palvelimilta voisi tarkastaa tärkeitä kohteita käyttämällä apuna NSCA:ta, NRPE:tä tai SNMP:tä. Tarkastettavia kohteita voisi olla esimerkiksi levytila, kuorma ja muut palvelimen ominaisuudet sekä palvelut. Myös postijonojen tarkastamiseksi on oma tarkastusohjelmansa *”check_mailq”*, joka ei kuitenkaan sovellu suoraan kaikille postiohjelmille. Nagioksen *”check_ldap*”-tarkastusohjelmalla voi tarkastaa käyttäjätunnusten hallinnassa käytetyn LDAPin toiminnan. MySQL:n tietokantojen toiminnan tarkastamiseksi on kehitetty *”check_mysql”*-tarkastusohjelma.

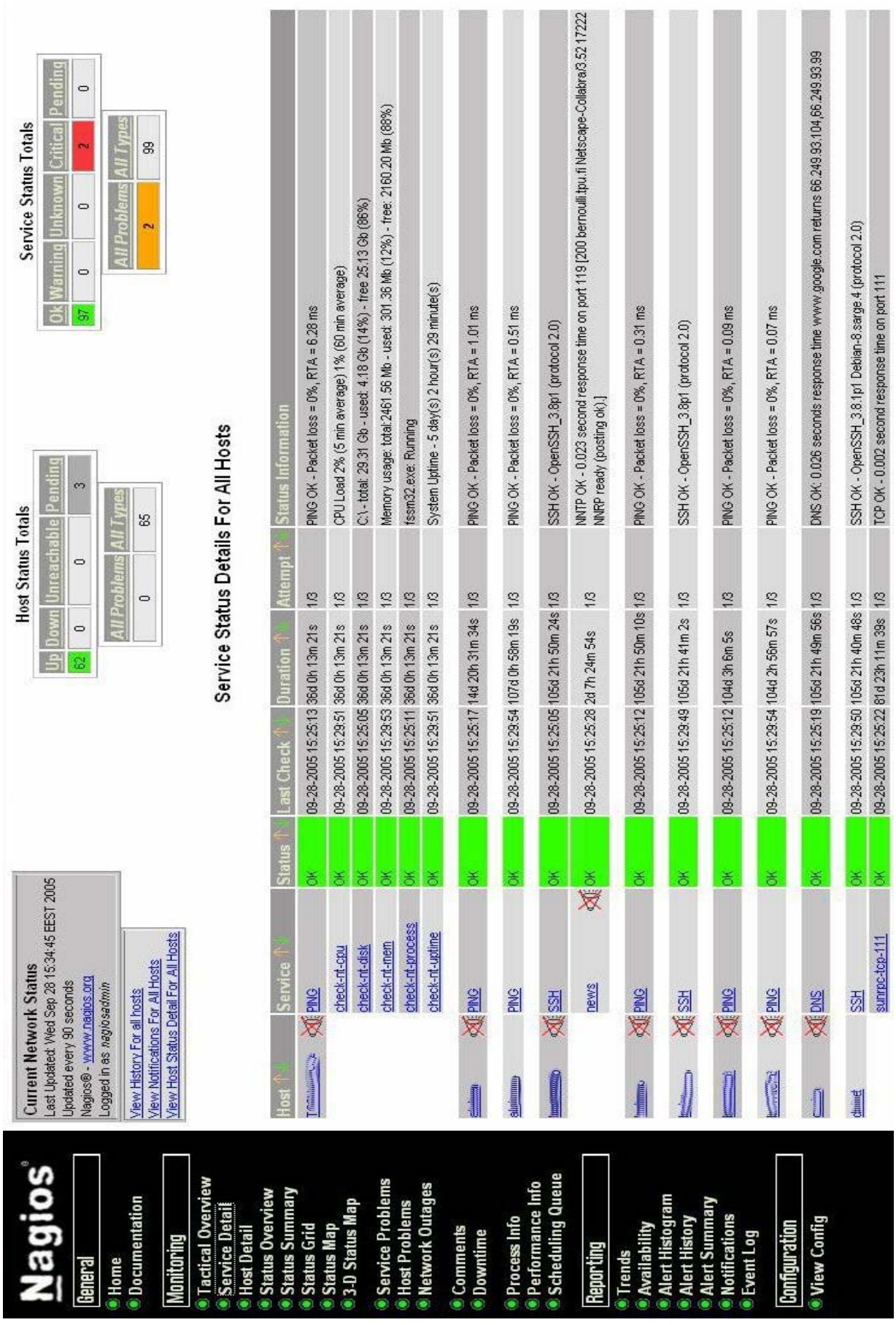
LÄHTEET

1 Ethan Galstad, [www-sivu]. [viitattu 10.11.2005] Saatavissa: http://www.nagios.org

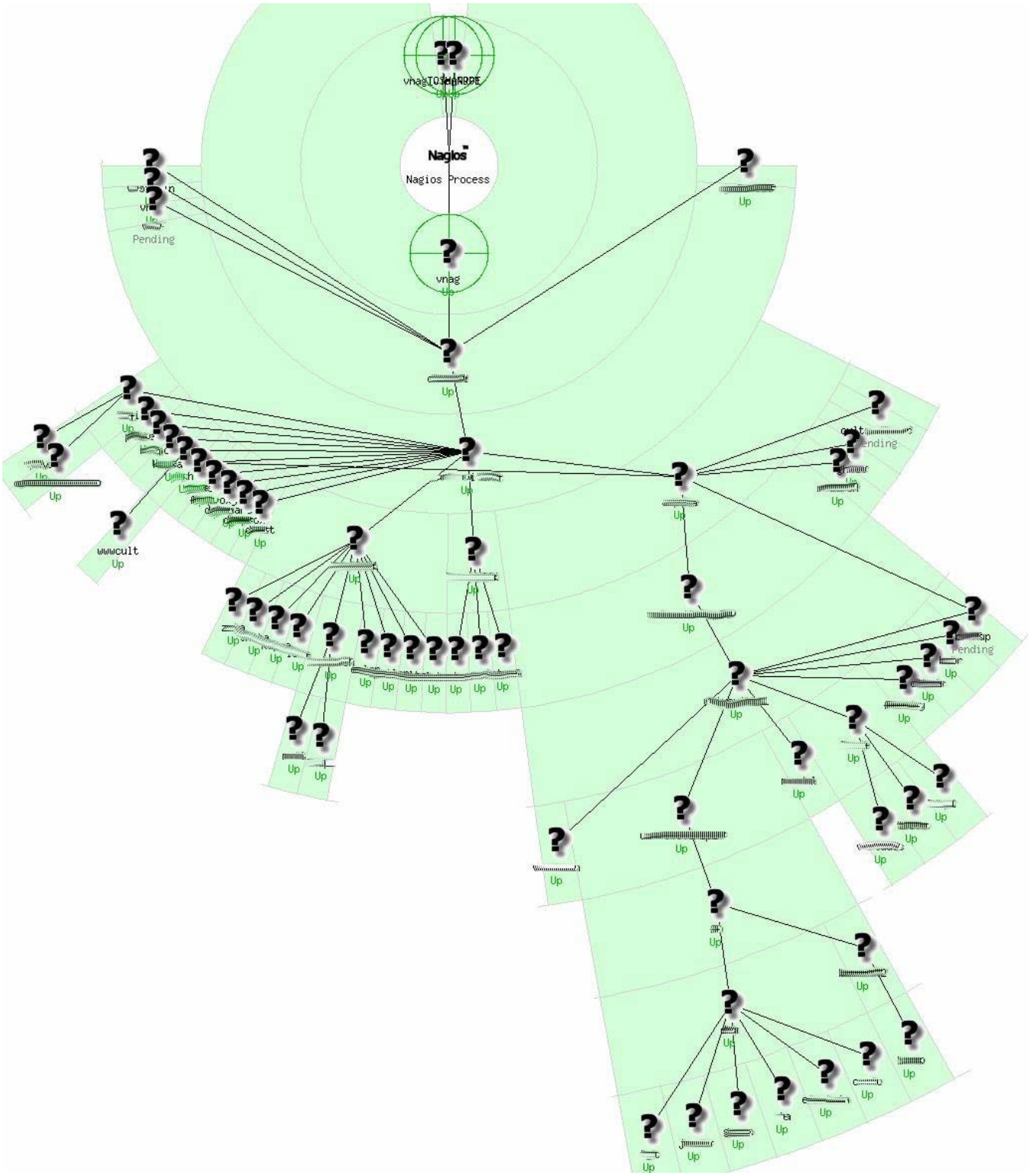
2 SourceForge, [www-sivu]. [viitattu 15.11.2005] Saatavissa: http://nagiosplug.sourceforge.net/developer-guidelines.html

3 NagiosExchange, [www-sivu]. [viitattu 27.11.2005] Saatavissa: http://www.nagiosexchange.org/

Liite 1: Palveluiden yhteenveto



Liite 2: Palvelinkartta



Liite 3: Uudet kohdat Nagios 2.0 konfiguraatiotiedostoissa

hosts.cfg

hostgroups=<palvelinryhmän_nimi>
: Määritellään mihin palvelinryhmään palvelin kuuluu.

check_interval=<aika>
: Voidaan määritellä palvelimen tarkastuksien suorittamiselle väli. Tätä ei kannata kuitenkaan tehdä jos se ei ole täysin pakollista, sillä palvelimien tila tarkastetaan normaalisti tarvittaessa.

active_checks_enabled=<0/1>
: Määritellään ovatko aktiiviset tarkastukset käytössä palvelimelle.

passive_checks_enabled=<0/1>
: Määritellään ovatko passiiviset tarkastukset käytössä palvelimelle.

check_period=<aikajakson_nimi>
: Määritellään aikajaksot jona palvelimen tarkastuksia suoritetaan.

obsess_over_host
: Määritellään suoritetaanko palvelimen tarkastusten suorituksen jälkeen pakonomaista ochp-komentoa.

check_freshness=<0/1> ja *freshness_threshold=<sekunteina>*
: Palvelimille voidaan ottaa tuoreuden tarkastaminen käyttöön. Tällä tarkastetaan tulevatko passiivisten tarkastusten tulokset ajallaan. Lisäksi määritellään raja-arvo palvelimen tuoreudelle.

contact_groups=<ryhmän_nimi>
: Määritellään ryhmät, joille ilmoituksia palvelimesta lähetetään. Ennen tämä määriteltiin palvelinryhmille.

services.cfg

servicegroups=<palveluryhmän_nimi>
: Myös palveluille voidaan määritellä ryhmiä.

contacts.cfg

contactgroups=<ryhmän_nimi>
: Määritellään ryhmä(t) joihin kontakti kuuluu.

addressx=<tieto>
: Kontaktille voidaan määritellä lisäosoitteita esimerkiksi puhelinnumero. Näitä voi olla kuusi kappaletta (address1…address6).

dependencies.cfg

inherits_parent=<0/1>
: Määritellään peritäänkö palvelulle tai riippuvuudet siltä palvelulta, josta ollaan riippuvaisia. Tämä määrittely on myös palvelimille.

escalations.cfg

escalation_period=<aikajakson_nimi>
: Määritellään aikajakso(t), jolloin ilmoitusten porrastusta käytetään. Voidaan määritellä sekä palveluille, että palvelimille.