Monitoraggio di sistemi

di Rudi Giacomini Pilon

Revisione 1.0

Questo documento tratterà di monitoraggio di sistemi e servizi distribuiti all'interno di una LAN o di una WAN. Ove possibile saranno portati, ad esempio, casi reali di implementazione.

Copyleft, Copyright, Limitazione di responsabilità

Buona lettura

Introduzione

Di solito quando si scrive un testo, sia esso un libro, una rivista o anche un semplice articolo, l'introduzione è l'ultima parte che viene scritta ma, a volte, è bello contravvenire alle regole. Questa introduzione viene scritta per prima e, mentre la scrivo, non ho idea se, alla fine, sarà un introduzione ad una serie di articoli, ad un manuale o ad un libro. Mi sono solo reso conto che su questo argomento c'è molto ancora da dire e da scrivere ed ho voluto fare la mia parte per tentare di colmare qualche lacuna..

In realtà quello che state per leggere è un insieme di appunti che verranno successivamente rivisti per dare loro una forma leggibile. Non credo che riuscirò a mantenere uno stile omogeneo: vi saranno punti in cui mi rivolgerò ad un principiante, in altri ad un sistemista esperto, in altri ancora ad un programmatore. In realtà sappiate che mi rivolgo anche a me stesso che sono un principiante quando affronto un nuovo problema, un sistemista con anni di esperienza alle spalle ed un programmatore per passione.

Perché allora scrivere un introduzione? Forse perché per una volta vorrei avere un obbiettivo più alto del solito e pensare che sto scrivendo per migliaia di persone, curiose come me di sapere come andrà a finire...

Il monitoraggio dei sistemi

Chiunque abbia avuto l' opportunità di amministrare una rete con un certo numero di PC, o comunque una rete in cui sono attivi dei servizi critici, ha certamente, prima o poi, avuto la necessità di monitorare gli elementi di criticità della stessa. Esistono infatti situazioni ove l'interruzione di un servizio può portare un grosso danno economico, commerciale o d'immagine. In altri casi, ancora più notevoli, un server bloccato o un servizio mancante può mettere a rischio delle persone.
Quasi tutti i grossi nomi del' informatica hanno a catalogo almeno una soluzione per tale necessità. Sono, solitamente, soluzioni che, a causa dei loro costi, trovano giustificazione di impiego solo dove la rete da amministrare sia veramente complessa o dove la criticità dei servizi sia molto elevata. Tali soluzioni sono quasi sempre notevolmente complesse da avviare e, in seguito, da utilizzare e per mitigare (apparentemente) tale situazione portano, a volte, con se una serie di servizi aggiuntivi per aiutare gli amministratori di rete anche nella gestione della stessa.

Non potevano mancare ovviamente anche delle soluzioni "Open Source" al problema e dopo averne provate alcune la mia scelta è caduta su Nagios [http://www.nagios.org].

Nagios©

Questo programma è stato scritto da Ethan Galstad sulla base di un suo programma precedente, Netsaint, il quale aveva, a suo parere, dei difetti di architettura tali da richiedere la riscrittura del codice. Non contiene funzioni per la gestione della rete ma solo per il monitoraggio di apparecchiature e servizi. Ho volutamente scritto apparecchiature, e non computer, in quanto il programma è in grado di monitorare anche switch, router, stampanti e altro, oltre che servizi più o meno standard come HTTP, FTP, POP3 e similari.

Nagios è fortemente personalizzabile, in quanto i controlli avvengono tramite programmi esterni (plugins) e la disponibilità del codice sorgente facilità la scrittura di nuovi test, qualora non esistessero già. La scelta di non includere i test all'interno del cuore del programma è, a mio parere, molto valida in quanto rende il programma facilmente espandibile a nuove funzionalità e nello stesso tempo permette a Galstad ed ai suoi collaboratori di concentrarsi solo sul motore del programma stesso.

Una volta rilevato un problema in un dispositivo sotto monitoraggio il programma è in grado di segnalare i malfunzionamenti attraverso la sua interfaccia (Web), spedendo un e-mail ad un gruppo di persone definito in precedenza o addirittura di avvisarle attraverso un SMS. Tale fase, detta notifica, è anch'essa fortemente personalizzabile e può essere gestita con un qualsiasi programma esterno al motore di Nagios.

Di seguito vedremo come installare il programma ed alcuni esempi di funzionamento.

Scelta del pacchetto

Nella sezione download [http://www.nagios.org/download/] del sito di Nagios sono disponibili sia i pacchetti precompilati per Red Hat e Fedora GNU/Linux, sia i sorgenti da compilare. Sarebbe opportuno sempre ovviamente

l'utilizzo dei sorgenti anche per avere a disposizione il codice per eventuali modifiche.

I plugins per Nagios sono, come detto, sviluppati separatamente dal corpo principale del programma. E` consigliabile, quindi , di scaricare subito anche i plugins dal sito [http://sourceforge.net/projects/nagiosplug/].

Dal sito NagiosExchange [www.nagiosexchange.org] all'indirizzo [http://www.nagiosexchange.org/Communication.41.0.html?&tx_netnagext_pi1[p_view]=140] potete scaricare il demone NSCA utile per l'inoltro dei test passivi e da [http://www.nagiosexchange.org/NRPE.77.0.html] NRPE per l'esecuzione di test remoti. Lo stesso sito è una fonte inesauribile di plugins e suggerimenti.

Non sono indispensabili, ma si possono ottenere anche delle icone per abbellire l'interfaccia web. Sono facilmente rintracciabili nel sito NagiosEchange [http://www.nagiosexchange.org/Logos_and_Images.18.0.html].

Fra i prerequisiti del programma vi sono:

- Un server web installato e correttamente funzionante per l'interfaccia web (si consiglia Apache [http://www.apache.org]).
- La versione 1.6.3 o maggiore della libreria gd di Thomas Boutell's richiesta per la gestione della grafica nell'interfaccia web.
- Un minimo di conoscenza su come si installa un programma GNU-Linux (anche se cercherò di ovviare spiegando in dettaglio i vari passi).
- Una buona conoscenza dei protocolli Internet, a partire dal TCP/IP, e dei sistemi da testare.

Il programma gira solo su sistemi *NIX like e le note che seguono si riferiscono all'installazione della versione2.0b4 di Nagios in una distribuzione GNU/Linux Fedora Core 4. Per completezza di informazione riporto che è stato possibile utilizzare la versione precedente anche su una distribuzione Red Hat 7.2 con il solo aggiornamento del PERL ed una piccola modifica ai sorgenti.

***Nota**: Quest'ultimo è un esempio tipico della convenienza di utilizzare i sorgenti. Red Hat ha una gestione particolare della localizzazione dei programmi per cui, a volte, i programmi, in cui non viene specificata una variabile locale di default, non funzionano. Così accade anche per il comando check_pinng incluso nella versione 1.4 di Nagios. Per far funzionare il tutto è sufficiente editare check_ping.c e alla linea 74 modificare*
`setlocale (LC_ALL, "") in`
`setlocale (LC_ALL, "C")`
La modifica è banale ma impossibile senza i sorgenti.

Installazione

L'installazione è semplice, anche se non banale, e ben documentata nel manuale in linea. Chi ne abbia la possibilità tenga a disposizione il manuale in linea per tutta la durata dell'installazione per copiare direttamente i comandi dal manuale ad un terminale. Chi non volesse rimanere connesso sappia che dopo la aver scompattato i sorgenti il manuale è disponibile localmente in nagios-2.0b4/html/docs/

E' bene precisare che le operazioni di installazione vanno eseguite come utente root. Tale pratica è solitamente sconsigliata ma, in questo caso, non può essere evitata. Si suppone comunque che se dovete monitorare dei sistemi e installare un programma di monitoraggio sappiate muovervi con la dovuta cautela fra i file di sistema.

Scompattiamo i sorgenti i una cartella a nostra scelta:

```
#tar -zxvf nagios-2.0b4.tar.gz
```

Creiamo l'utente ed il gruppo per nagios e aggiungiamo al gruppo l'utente apache che deve avere accesso all'interfaccia web del programma. Il programma utilizzerà i permessi di questo utente/gruppo durante l'esecuzione aumentando il grado di sicurezza del sistema. Non sarebbe sicuro infatti se Nagios venisse fatto girare con i permessi di root in quanto in caso di exploit di Nagios a causa di una suo (sempre possibile) bug si riuscirebbe a guadagnare il controllo del sistema. Creando un utente ad-hoc il problema viene notevolmente ridotto.

```
# adduser nagios
# /usr/sbin/groupadd nagcmd
# /usr/sbin/usermod -G nagcmd apache
# /usr/sbin/usermod -G nagcmd nagios
```

Creiamo la directory che ospiterà il programma e settiamo i corretti permessi per l'utente appena creato.

```
# mkdir /usr/local/nagios
# chown nagios.nagios /usr/local/nagios
```

Benché sia possibile compilare il programma con varie opzioni relative ai percorsi di utilizzo, a meno che non vi siano rischi di sicurezza, si consiglia di non modificare le impostazioni standard in modo che le istruzioni seguenti siano coerenti. Quindi usiamo l'istruzione *configure* senza parametri

```
# cd nagios-2.0b4
# ./configure
------
[---si omette l'output, per brevità, riportando solo l'ultima parte---]
------
*** Configuration summary for nagios 2.0b4 08-02-2005 ***:

 General Options:
 -------------------------
        Nagios executable:  nagios
        Nagios user/group:  nagios,nagios
       Command user/group:  nagios,nagios
            Embedded Perl:  no
             Event Broker:  yes
        Install ${prefix}:  /usr/local/nagios
                Lock file:  ${prefix}/var/nagios.lock
           Init directory:  /etc/rc.d/init.d
                  Host OS:  linux-gnu

 Web Interface Options:
 ------------------------
                HTML URL:  http://localhost/nagios/
                 CGI URL:  http://localhost/nagios/cgi-bin/
 Traceroute (used by WAP):  /bin/traceroute


Review the options above for accuracy.  If they look okay,
type 'make all' to compile the main program and CGIs.
```

Il risultato dell'operazione di configurazione dovrebbe riportare tutti i percorsi configurati. Può essere utile prendere nota di tali dati e poi passare a compilare ed installare:

```
# make all
# make install
# make install-init
# make install-commandmode
# make install-config
```

L'ultima istruzione installa i file di esempio di configurazione che ci serviranno poi per avere una base da cui partire senza riscriverli da zero. A questo punto è possibile passare all'installazione dei plugins che vanno scaricati e compilati separatamente e alla configurazione dell'interfaccia web. Le operazioni sono le seguenti:

```
# tar -zxvf nagios-plugins-1.4.2.tar.gz
# cd nagios-plugins-1.4.2
# ./configure
# make
# make install
```

La configurazione dell'interfaccia web si basa sull'ipotesi che utilizziate il server Apache e sia installato nella stessa macchina in cui è installato Nagios. La corretta configurazione di Apache è, chiaramente, un prerequisito e le note che seguono servono per la sola configurazione di Nagios.

Nel nostro esempio le modifiche vanno applicate al file di configurazione di Apache ovvero:
```
 # vi /etc/httpd/conf/httpd.conf
```
alla fine del file, come da manuale di Nagios, aggiungiamo quanto segue:

```
ScriptAlias /nagios/cgi-bin /usr/local/nagios/sbin
<Directory "/usr/local/nagios/sbin">
    AllowOverride AuthConfig
    Options ExecCGI
    Order allow,deny
    Allow from all
</Directory>

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

<Directory /usr/local/nagios/sbin>
AllowOverride AuthConfig
order allow,deny
allow from all
Options ExecCGI
</Directory>

<Directory /usr/local/nagios/share>
AllowOverride AuthConfig
order allow,deny
allow from all
</Directory>
```

Fra le varie istruzioni qui sopra vi è quella di utilizzare dei file locali per la verifica degli accessi. Quindi è necessario configurare un file di accesso come segue:

```
# cd /usr/local/nagios/sbin
# vi  .htaccess
```

ed inserire nel file quanto segue:

```
AuthName "Nagios Access"
AuthType Basic
AuthUserFile /usr/local/nagios/etc/htpasswd.users
require valid-user
```

Creiamo quindi i permessi per un paio di utenti:

```
# cd /usr/local/nagios/etc
# htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password: ***
Re-type new password: ***
Adding password for user nagiosadmin
```

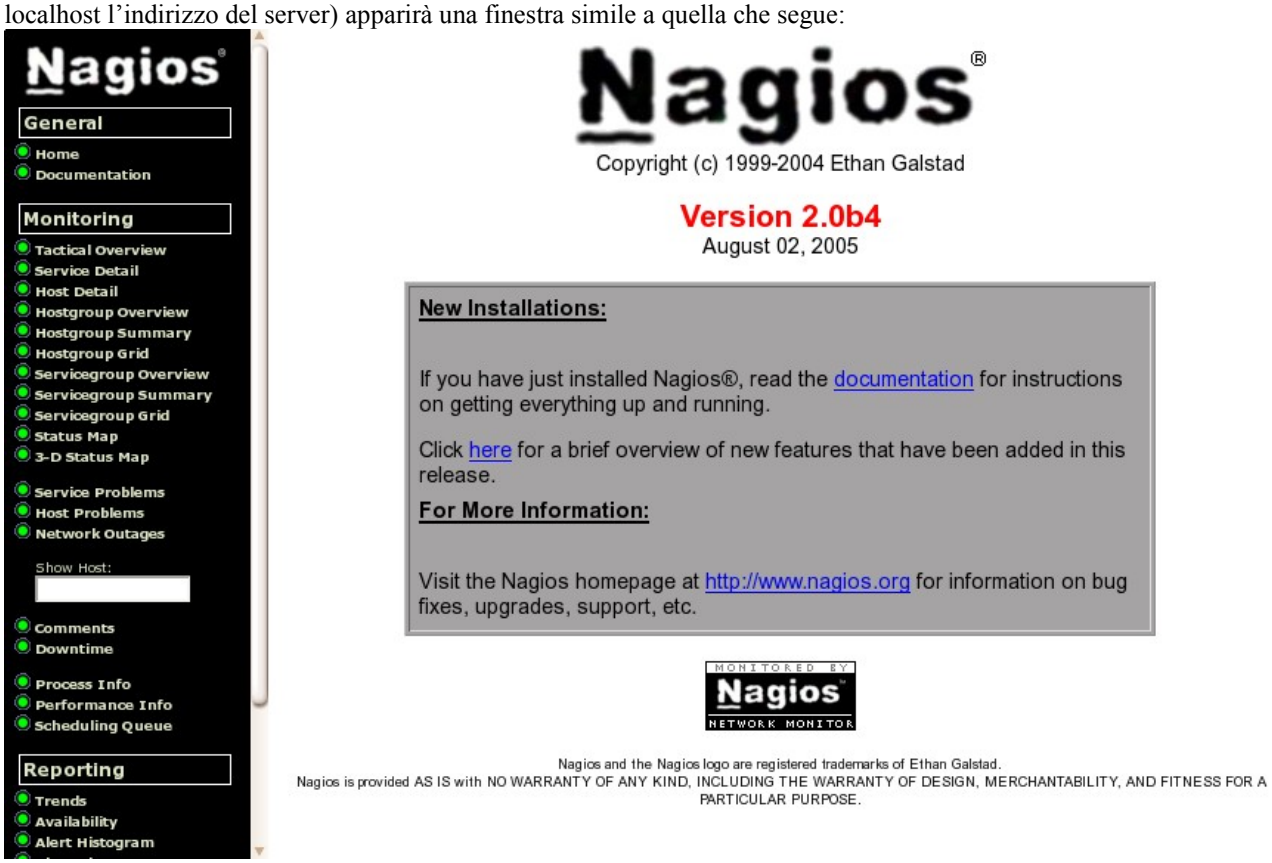
Aggiungete anche il vostro account abituale (negli esempi sarà **rudig**)

```
# htpasswd /usr/local/nagios/etc/htpasswd.users rudig
New password:
Re-type new password:
Adding password for user rudig
```

Ovviamente va aggiunto il dovuto permesso per ogni amministratore o utente che deve accedere all'interfaccia web di

Nagios.

A questo punto si può considerare completata l'installazione e digitando nel browser al percorso http://localhost/nagios/ (se utilizzate il browser in una macchina diversa da quella che contiene Nagios bisognerà ovviamente sostituire a localhost l'indirizzo del server) apparirà una finestra simile a quella che segue:



chiaramente selezionando le varie voci dei menù si otterranno solo degli errori, ma se si riuscirà a visualizzare questa pagina si potrà passare alla fase di configurazione.

Configurazione di base

Inizialmente si preparano i file di configurazione partendo dai file di esempio. In seguito ottimizzeremo questa struttura per avere una migliore gestione.

```
# cd /usr/local/nagios/etc/
# cp nagios.cfg-sample nagios.cfg
# cp checkcommands.cfg-sample checkcommands.cfg
# cp resource.cfg-sample resource.cfg
# cp misccommands.cfg-sample misccommands.cfg
# cp cgi.cfg-sample cgi.cfg
# cp minimal.cfg-sample minimal.cfg
```

Il file di base della configurazione (nagios.cfg) contiene dei riferimenti a dei file esterni che vengono inclusi. Nella configurazione di esempio il file incluso che andremo a modificare è uno solo (minimal.cfg).

Iniziamo quindi a modificare il file minimal.cfg:

All'inizio del file vengono definiti i Time Periods ovvero le finestre temporali nelle quali verranno gestiti gli eventi o spedite le notifiche. Il file definisce solo il periodo "24x7" ovvero tutte le ventiquattro ore di tutti i giorni della

settimana. Per iniziare con il piede giusto consiglio di definire altri due periodi, uno relativo alle ore lavorative, l'altro relativo alle ore non lavorative. Se nel vostro caso si tratta di turni di lavoro nelle 24 ore, o simili, può essere utile definire tali turni.

Quindi al termine della definizione del periodo 24x7 aggiungiamo altri due periodi più un periodo per definire un periodo vuoto utile per indicare la non esecuzione di un evento o altre finezze del genere:

```
# '24x7' timeperiod definition
define timeperiod{
        timeperiod_name 24x7
        alias           24 Hours A Day, 7 Days A Week
        sunday          00:00-24:00
        monday          00:00-24:00
        tuesday         00:00-24:00
        wednesday       00:00-24:00
        thursday        00:00-24:00
        friday          00:00-24:00
        saturday        00:00-24:00
        }


# 'workhours' timeperiod definition
define timeperiod{
        timeperiod_name workhours
        alias           "Normal" Working Hours
        monday          08:00-18:00
        tuesday         08:00-18:00
        wednesday       08:00-18:00
        thursday        08:00-18:00
        friday          08:00-18:00
        }

# 'nonworkhours' timeperiod definition
define timeperiod{
        timeperiod_name nonworkhours
        alias           Non-Working Hours
        sunday          00:00-24:00
        monday          00:00-08:00,18:00-24:00
        tuesday         00:00-08:00,18:00-24:00
        wednesday       00:00-08:00,18:00-24:00
        thursday        00:00-08:00,18:00-24:00
        friday          00:00-08:00,18:00-24:00
        saturday        00:00-24:00
        }


# 'none' timeperiod definition
define timeperiod{
        timeperiod_name none
        alias           No Time Is A Good Time
        }
```

Passiamo ora direttamente alla modifica dei contatti ovvero l'elenco delle persone che verranno contattate ogni volta che viene rilevato un problema. Da notare che Nagios permette di definire dei contatti diversi per ogni macchina, gruppo di macchine, servizio, gruppi di servizi. Nell'esempio che segue inserirò un solo nominativo come contatto, quindi, eliminate i contatti standard inseriti e inserite i vostri riferimenti come da esempio:

```
# 'rudig' contact definition
define contact{
        contact_name                    rudig              #nome dell'account
        alias                           Rudi Giacomini     #nome esteso
        service_notification_period     workhours #si vuole ricevere notifiche
```

```
        host_notification_period        workhours #solo durante l'orario di
lavoro
        service_notification_options    c,r
        host_notification_options       d,r
        service_notification_commands   notify-by-email      #riceveremo la
        host_notification_commands      host-notify-by-email #notifica via email
        email                           rudig@localhost
        }
```

Come si vede la definizione del precedente periodo temporale "workhours" è già utile in quanto impiegata per indicare gli orari nei quali si desidera ricevere la notifica. Per quanto riguarda le voci **service_notification_options** e **host_notification_options** vengono utilizzate per indicare quali stati del sistema sotto controllo si desidera vengano notificati. I valori possibili sono: u=unreachable (irragiungibile), d= down (spento-assente), r=recoveries (ripristino del servizio), f=flapping (intermittente-instabile), w=warning (avvisi), c=critical (condizione critica-guasto), n=none (nessuna segnalazione).

Modificate anche i gruppi dei contatti inserendo i nominativi aggiunti come da esempio:

```
define contactgroup{
        contactgroup_name       admins
        alias                   Nagios Administrators
        members                 rudig
        }
```

Nel nostro esempio è sufficiente un gruppo, ma se gestite un'azienda molto ampia con personale molto specializzato potreste definire un gruppo da contattare per i problemi relativi a Linux, un' altro gruppo per i database, uno per il gestionale e così via. Potrebbe essere importane anche la definizione di contatti o gruppi di contatti "geograficamente" vicini ai dispositivi da controllare; cosa ancora più vera nel caso di reti distribuite sul territorio.

Un passo ulteriore: Cancellate la definizione dei comandi dal file minimal.cfg. Cancellate tutta la sezione. I comandi riportati in tale file sono un duplicato di quelli del file checkcommands.cfg che viene, giustamente, usato dal file nagios.cfg.

Ora la parte più importante ovvero la definizione di cosa controllare. Seguiamo per un attimo il file di esempio e decidiamo di mettere sotto controllo l' host che esegue Nagios stesso.
Modifichiamo leggermente l' host ed il gruppo di default ottenendo quanto segue:

```
define host{
        use                 generic-host  ;ci basiamo su un template generico
predefinito
        host_name           localhost
                            alias         Nagios Server
        address               127.0.0.1   ;indirizzo per il momento usiamo
localhost
        check_command  check-host-alive   ; tipo di test da eseguire
        max_check_attempts      10              ;tentativi massimi
        notification_interval   120
        notification_period     24x7
        notification_options    d,r
        contact_groups          admins
        }


# We only have one host in our simple config file, so there is no need to
# create more than one hostgroup.

define hostgroup{
        hostgroup_name  test
```

```
        alias           Primo test
        members         localhost
        }
```

Per quanto riguarda i servizi sotto test manteniamo inalterato quanto proposto nei files di esempio.

Ultimo sforzo: sistemiamo i permessi per l'interfaccia grafica editando il file cgi.cfg individuate tutte le righe che iniziano con authorized_for_ rimuovete il commento e inserite alla fine il nome che avete definito quando avete configurato gli accessi alla sezione web.
La riga di esempio dovrebbe chiarire più di mille parole:

```
authorized_for_system_information=admin,nagios,rudig
```

Dopo aver preparato la configurazione Nagios mette a disposizione un comando per verificare la stessa:

```
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

E` possibile preparare uno script per semplificare il test, sia per evitare di digitare ogni volta tutta la sintassi sia perché sia il comando sia il file da testare saranno sempre gli stessi, in quanto, partendo dal file nagios.cfg, il comando di test è in grado di testare tutti i file dipendenti inclusi a partire da questo. Quindi con
`# vi /usr/bin/chechnagios`

inserendo le righe seguenti:

```
#!/bin/sh
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

e poi settando il file come eseguibile: `# chmod +x /usr/local/bin/chechnagios`

si può eseguire il comando. Quindi ora è possibile testare il file di configurazione:
`# checknagios`

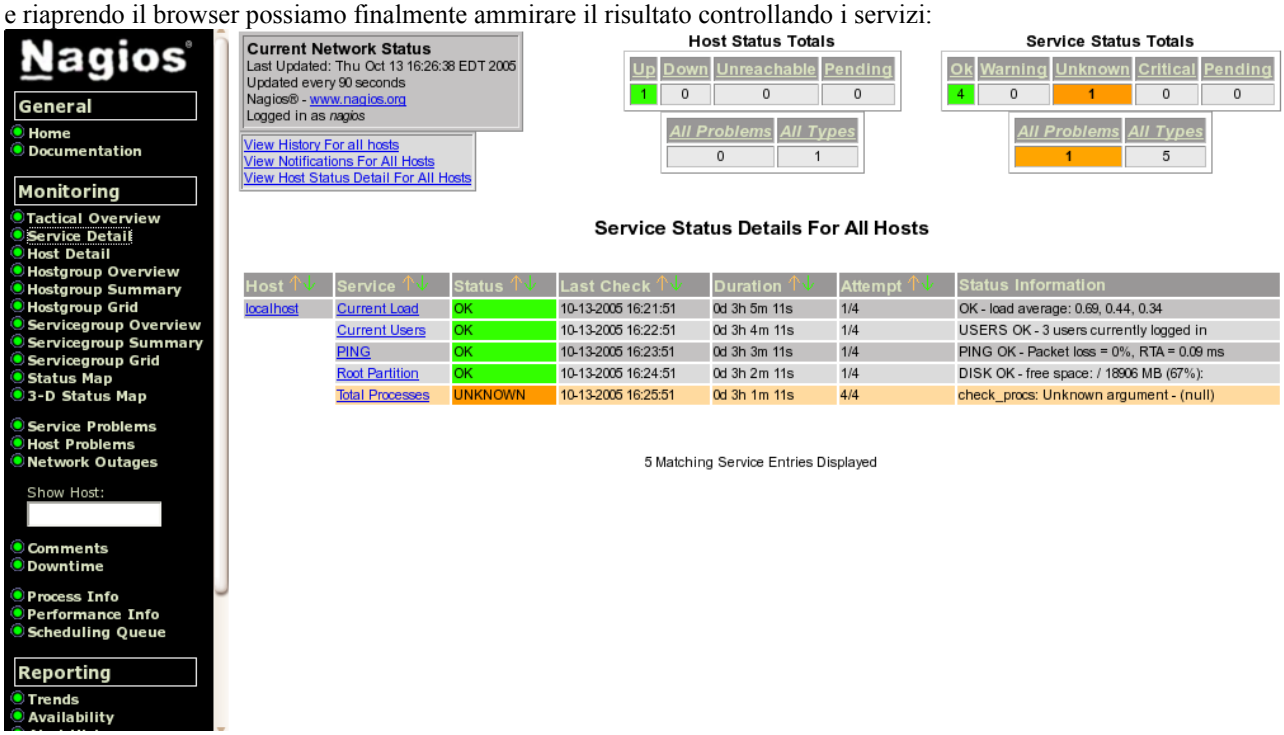
```
Nagios 2.0b4
Copyright (c) 1999-2005 Ethan Galstad (http://www.nagios.org)
Last Modified: 08-02-2005
License: GPL

Reading configuration data...

Running pre-flight check on configuration data...

Checking services...
        Checked 5 services.
Checking hosts...
        Checked 1 hosts.
Checking host groups...
        Checked 1 host groups.
Checking service groups...
        Checked 0 service groups.
Checking contacts...
        Checked 1 contacts.
Checking contact groups...
        Checked 1 contact groups.
Checking service escalations...
        Checked 0 service escalations.
Checking service dependencies...
        Checked 0 service dependencies.
Checking host escalations...
        Checked 0 host escalations.
Checking host dependencies...
        Checked 0 host dependencies.
Checking commands...
```

```
        Checked 22 commands.
Checking time periods...
        Checked 4 time periods.
Checking extended host info definitions...
        Checked 1 extended host info definitions.
Checking extended service info definitions...
        Checked 0 extended service info definitions.
Checking for circular paths between hosts...
Checking for circular host and service dependencies...
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
```

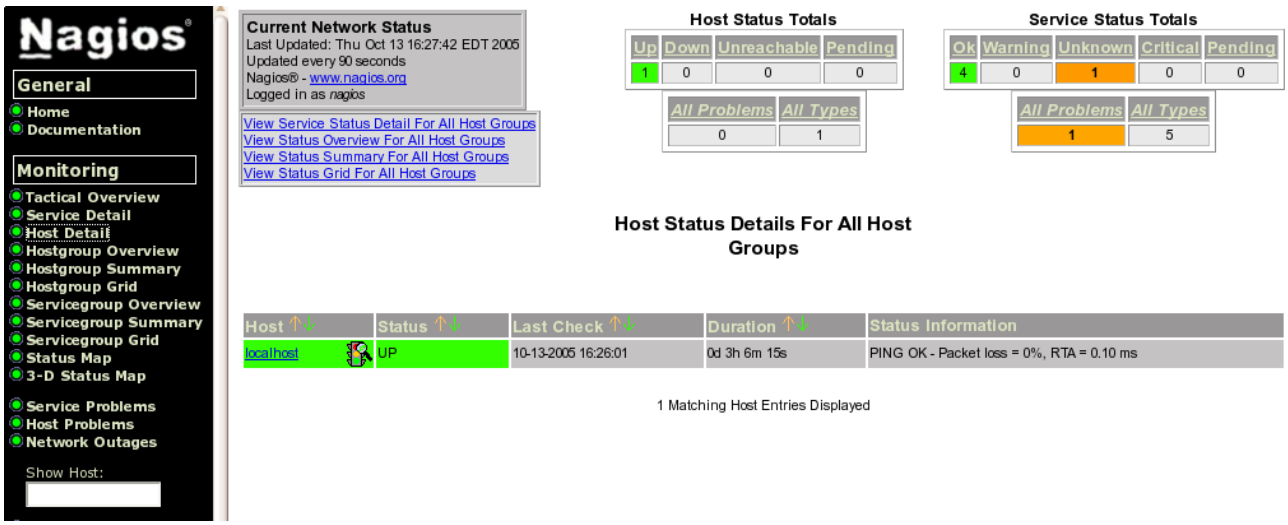
come si vede va tutto bene e possiamo verificare il sistema facendo partire il demone di controllo:

```
# service nagios start
```

e riaprendo il browser possiamo finalmente ammirare il risultato controllando i servizi:



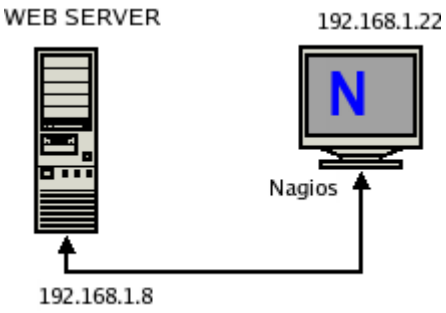
Come si vede dal' immagine oltretutto nella configurazione di default c'è un errore nella definizione di un servizio che, quindi, risulta come un problema.



Poiché si voleva solo eseguire un test del sistema è possibile ignorare l'errore relativo al comando check_procs e passare a qualcosa di più reale.

Esempio di controllo di un server

Passiamo ad un esempio un po più serio e supponiamo di dover controllare un server che ospita un servizio web con un database MySQL e un mail server. Supponiamo che il server Nagios abbia indirizzo 192.168.1.22 e il Web server 192.168.1.8 e che i due host si trovino sulla stessa rete locale.



Per iniziare a migliorare la gestione manteniamo il file minimal.cfg che conterrà, invariate, le definizione dei periodi temporali, dei contatti e dei gruppi di contatti. Separiamo le definizione degli hosts e dei servizi in due file aggiuntivi.

Modifichiamo quindi per primo nagios.cfg rimuovendo il commento dalle righe relative a hosts e services:

#cfg_file=/usr/local/nagios/etc/hostgroups.cfg

cfg_file=/usr/local/nagios/etc/hosts.cfg

cfg_file=/usr/local/nagios/etc/services.cfg

#cfg_file=/usr/local/nagios/etc/timeperiods.cfg

rimuoviamo da minimal.cfg qualsiasi riferimento a host e servizi e andiamo a preparare hosts.cfg

Prima di inserire le configurazioni degli host e dei servizi chiariamo un concetto sul modello di definizione usato da Nagios: Nagios permette l'uso dei template (modelli) per le definizione di host e servizi. Si possono cioè inserire delle descrizioni generiche con già preconfigurati tutti i parametri standard. Nella definizione del singolo host o servizio si può indicare di utilizzare il modello e poi aggiungere solo i parametri mancanti o quelli che variano rispetto al modello.

Fra l'altro è permesso l'uso di modelli in cascata cosa che sarà immediatamente sfruttata e descritta:

Creiamo il file hosts.cfg e inseriamo il template generico di host già presente nel file minimal.cfg.

```
##############################################################################
# HOST DEFINITIONS
##############################################################################

# Generic host definition template
define host{
        name      generic-host    ; il nome di questo template
      notifications_enabled   1       ; Notifiche abilitate
      event_handler_enabled   1       ; Eventi abilitati
      flap_detection_enabled  1       ; Rilevamento stati indefiniti abilitato
      process_perf_data       1       ; Analisi performance abilitata
      retain_status_information      1       ; Mantenimento stato d'errore
      retain_nonstatus_information    1      ; Mantenimento informazioni
aggiuntive
       register               0       ; NON VA` REGISTRATO NON E` UN HOST REALE.
      }
```

Da questo ereditiamo subito un modello più dettagliato:

```
# Template più dettagliato
define host{
        use       generic-host            ; Eredita il template precedente
        name      my_host
        check_command           check-host-alive     ;Test di default
        max_check_attempts      10                   ;Massimo num. tentativi
        notification_interval   120                  ;Intervallo di notifica
        notification_period     24x7                 ;Riferimento al periodo
        notification_options    d,r                  ; Eventi da notificare
        contact_groups          admins               ;Gruppo da contattare
        register                0     ; NON VA` REGISTRATO NON E` UN HOST REALE.
        }
```

Definiamo ora il server da controllare:

```
# 'web_server' definizione
define host{
        use             my_host                 ; Eredita i modelli
        host_name       web_server              ;nome del server
        alias           Linux Web & Mail
        address         192.168.1.8             ;indirizzo
        }

#informazioni aggiuntive
define hostextinfo {
        host_name       web_server
        icon_image      linux.png
        icon_image_alt  Linux Host
        vrml_image      linux.png
        statusmap_image linux.gd2
        }
```

Nelle informazioni aggiuntive sono stati inseriti solo i riferimenti alle icone da usare.

E ora il gruppo (anche se per ora abbiamo un solo host)

```
##############################################################################
# HOST GROUP DEFINITIONS
##############################################################################
# 'linux-boxes' host group
define hostgroup{
        hostgroup_name  linux-boxes
```

```
        alias           Linux Servers
        members         web_server
        }
```

Passiamo quindi alla definizione dei servizi nel file services.cfg:

```
###############################################################################
# SERVICE DEFINITIONS
###############################################################################

#  template per un servizio generico
define service{
        name                            generic-service    ; Nome del template
        active_checks_enabled       1     ; Controllo di tipo attivo abil.
        passive_checks_enabled      1     ; Controllo passivo abilitato
        parallelize_check           1     ; Attiva controlli in parallelo
        obsess_over_service          1    ;Se necessario mantiene il
monitoraggio
        check_freshness             0     ; Non controlla se il dato è fresco
        notifications_enabled       1     ; Notifiche abilitate
        event_handler_enabled       1     ; Abilita la gestione del servizio
        flap_detection_enabled      1     ; Abilita su stato instabile
        process_perf_data           1     ; Abilita controllo performances
        retain_status_information   1     ; Mantiene le informazioni su riavvio
        retain_nonstatus_information    1
        register                        0  ; NON REGISTRA IL SERVIZIO (TEMPLATE)
        }

# Test del mail server
define service{
        use                     generic-service   ; Usa il template precedente
        host_name               web_server        ;nome server
        service_description     SMTP              ;nome servizio
        is_volatile             0                 ;non è volatile
        check_period            24x7              ;periodo usato per i test
        max_check_attempts      3                 ;massimo numero di tentativi
        normal_check_interval   3                 ;intervallo fra i test
        retry_check_interval    1                 ;intervallo in caso di errore
        contact_groups          admins            ;contatti
        notification_interval   120               ;intervallo fra le notifiche
        notification_period     24x7              ;periodo di notifica
        notification_options    w,u,c,r           ;errori notificati
        check_command           check_smtp        ;comando usato per i test
        }

# Test del web server
define service{
        use                     generic-service   ; Usa il template precedente
        host_name               web_server        ;nome server
        service_description     HTTP              ;nome servizio
        is_volatile             0                 ;non è volatile
        check_period            24x7              ;periodo usato per i test
        max_check_attempts      3                 ;massimo numero di tentativi
        normal_check_interval   3                 ;intervallo fra i test
        retry_check_interval    1                 ;intervallo in caso di errore
        contact_groups          admins            ;contatti
        notification_interval   120               ;intervallo fra le notifiche
        notification_period     24x7              ;periodo di notifica
        notification_options    w,u,c,r           ;errori notificati *
        check_command           check_http
        }
```

**Per quel che riguarda gli errori notificati le opzioni sono: w=warning (avvisi), c=critical (errori critici), u=unknown (sconosciuto) e r=recoveries (riavvio) f=flapping(instabile) n=none (nessuna segnalazione).*

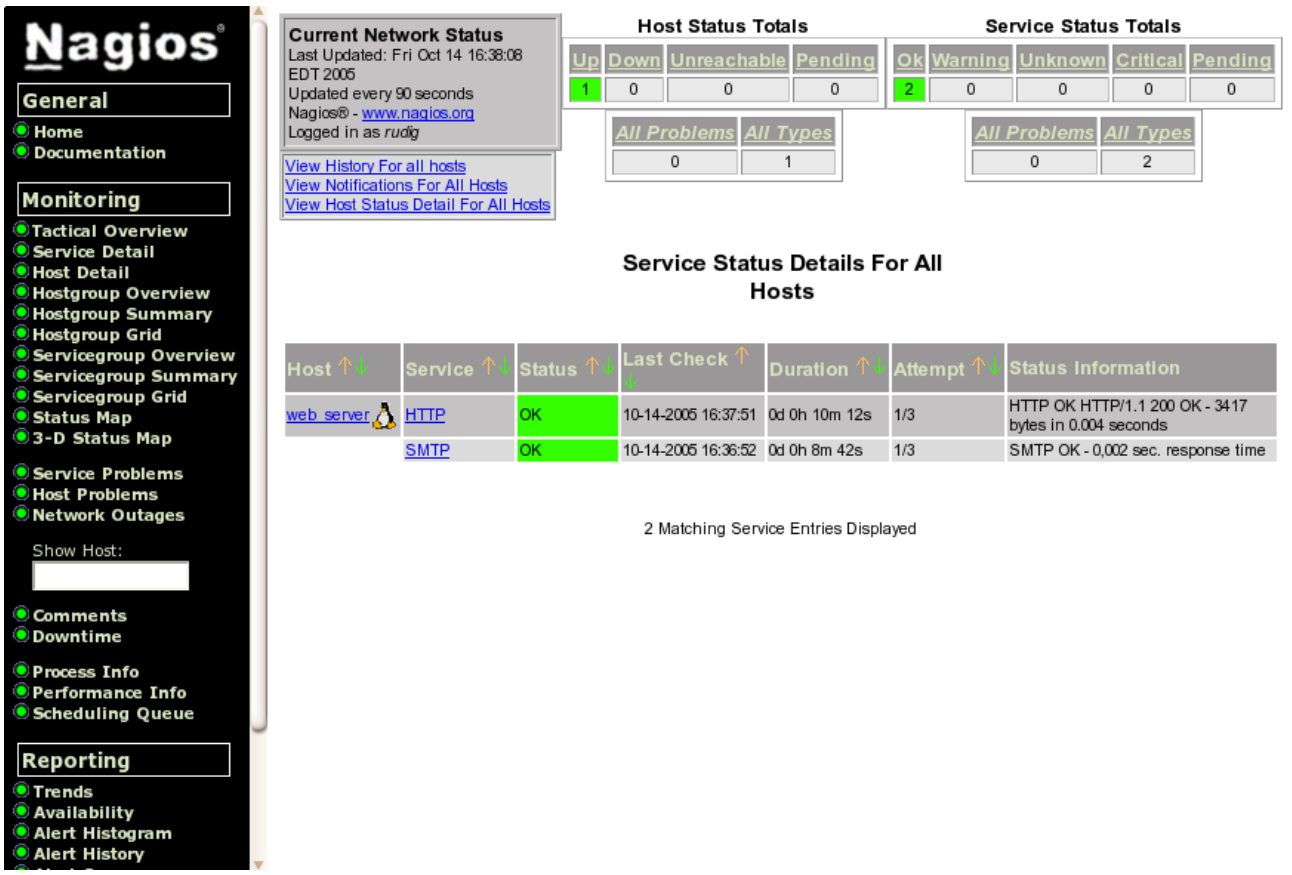
Abbiamo definito due dei tre servizi che ci eravamo prefissi di controllare. In caso di errore di un servizio verrà inviata una notifica al gruppo indicato.
Testiamo la nuova configurazione con il comando

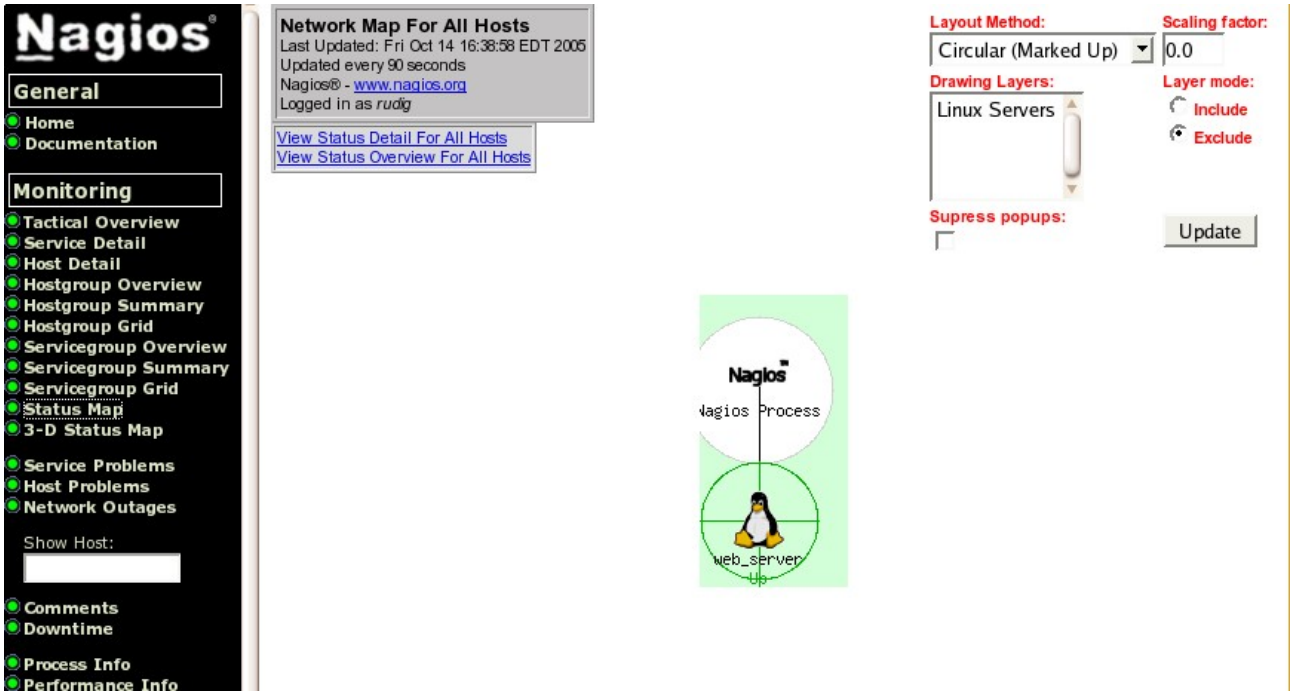
```
# checknagios

Nagios 2.0b4
Copyright (c) 1999-2005 Ethan Galstad (http://www.nagios.org)
Last Modified: 08-02-2005
License: GPL

Reading configuration data...

Running pre-flight check on configuration data...

Checking services...
        Checked 2 services.
Checking hosts...
        Checked 1 hosts.
Checking host groups...
        Checked 1 host groups.
Checking service groups...
        Checked 0 service groups.
Checking contacts...
        Checked 1 contacts.
Checking contact groups...
        Checked 1 contact groups.
Checking service escalations...
        Checked 0 service escalations.
Checking service dependencies...
        Checked 0 service dependencies.
Checking host escalations...
        Checked 0 host escalations.
Checking host dependencies...
        Checked 0 host dependencies.
Checking commands...
        Checked 22 commands.
Checking time periods...
        Checked 4 time periods.
Checking extended host info definitions...
        Checked 1 extended host info definitions.
Checking extended service info definitions...
        Checked 0 extended service info definitions.
Checking for circular paths between hosts...
Checking for circular host and service dependencies...
Checking global event handlers...
Checking obsessive compulsive processor commands...
Checking misc settings...

Total Warnings: 0
Total Errors:   0

Things look okay - No serious problems were detected during the pre-flight check
```

La verifica ci conferma che abbiamo definito un host e due servizi per cui facciamo ripartire Nagios

```
# service nagios restart
```

e apriamo l'interfaccia web per verificare:





Non male come primo risultato ma dobbiamo ancora testare il database.

Per il test del database MySQL dobbiamo modificare i comandi standard aggiungendo un comando ad-hoc. Nella directory `/usr/local/nagios/libexec` troviamo il comando `check_mysql` ed eseguendolo col l'opzione `--help` otteniamo

```
# ./check_mysql --help
check_mysql (nagios-plugins 1.4.2) 1.26
Copyright (c) 1999-2004 Nagios Plugin Development Team
        <nagiosplug-devel@lists.sourceforge.net>

This program tests connections to a mysql server
Usage: check_mysql [-d database] [-H host] [-P port] [-u user] [-p password] [-
S]

Options:
 -h, --help
    Print detailed help screen
 -V, --version
    Print version information
 -H, --hostname=ADDRESS
    Host name or IP Address
 -P, --port=INTEGER
    Port number (default: 3306)
 -d, --database=STRING
   Check database with indicated name
 -u, --username=STRING
   Connect using the indicated username
 -p, --password=STRING
   Use the indicated password to authenticate the connection
   ==> IMPORTANT: THIS FORM OF AUTHENTICATION IS NOT SECURE!!! <==
   Your clear-text password will be visible as a process table entry
 -S, --check-slave
   Check if the slave thread is running properly.

There are no required arguments. By default, the local database with
a server listening on MySQL standard port 3306 will be checked

Send email to nagios-users@lists.sourceforge.net if you have questions
regarding use of this software. To submit patches or suggest improvements,
send email to nagiosplug-devel@lists.sourceforge.net
```

Notate che viene chiaramente indicato che la password utilizzata sarà visibile in chiaro per cui non utilizzate questo test in un ambiente a rischio. Modifichiamo, quindi, il file checkcommands.cfg aggiungendo banalmente:

```
# 'check_mysql' command definition
define command{
        command_name    check_mysql
        command_line    $USER1$/check_mysql -d my_db -H 192.168.1.4  -u rudig -p
testpass
        }
```

e poi il file services.cfg aggiungendo:

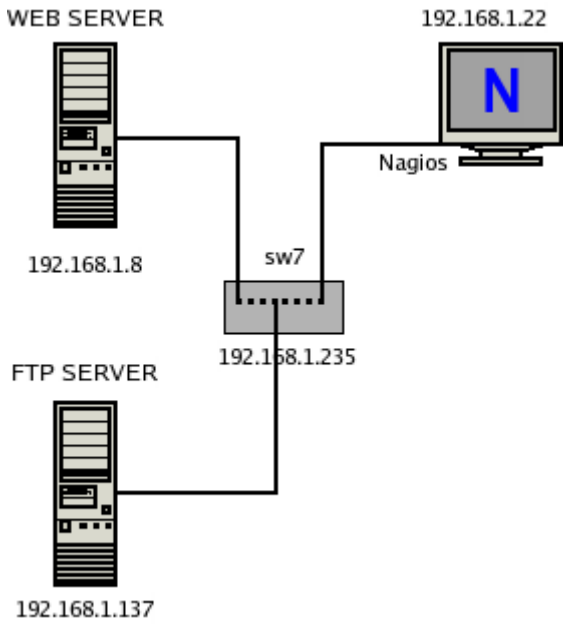
```
# Test del database
define service{
        use                     generic-service ; Usa il template precedente
        host_name               web_server              ;nome server
        service_description     MYSQL                   ;nome servizio
        is_volatile             0                 ;non è volatile
        check_period            24x7                    ;periodo usato per i test
        max_check_attempts      3                 ;massimo numero di tentativi
        normal_check_interval   3                 ;intervallo fra i test
        retry_check_interval    1                 ;intervallo in caso di errore
        contact_groups          admins            ;contatti
        notification_interval   120               ;intervallo fra le notifiche
        notification_period     24x7              ;periodo di notifica
        notification_options    w,u,c,r           ;errori notificati
```

```
        check_command           check_mysql
        }
```

Dopo il solito riavvio otterremo il risultato voluto.

Una rete più complessa...

Penso sia lecito ipotizzare che la vostra rete sia un po più complessa. Per non strafare ipotizziamo un FTP server all'indirizzo 192.168.1.137 come da schema.



aggiungiamo quindi l' host ed il servizio nei relativi files:

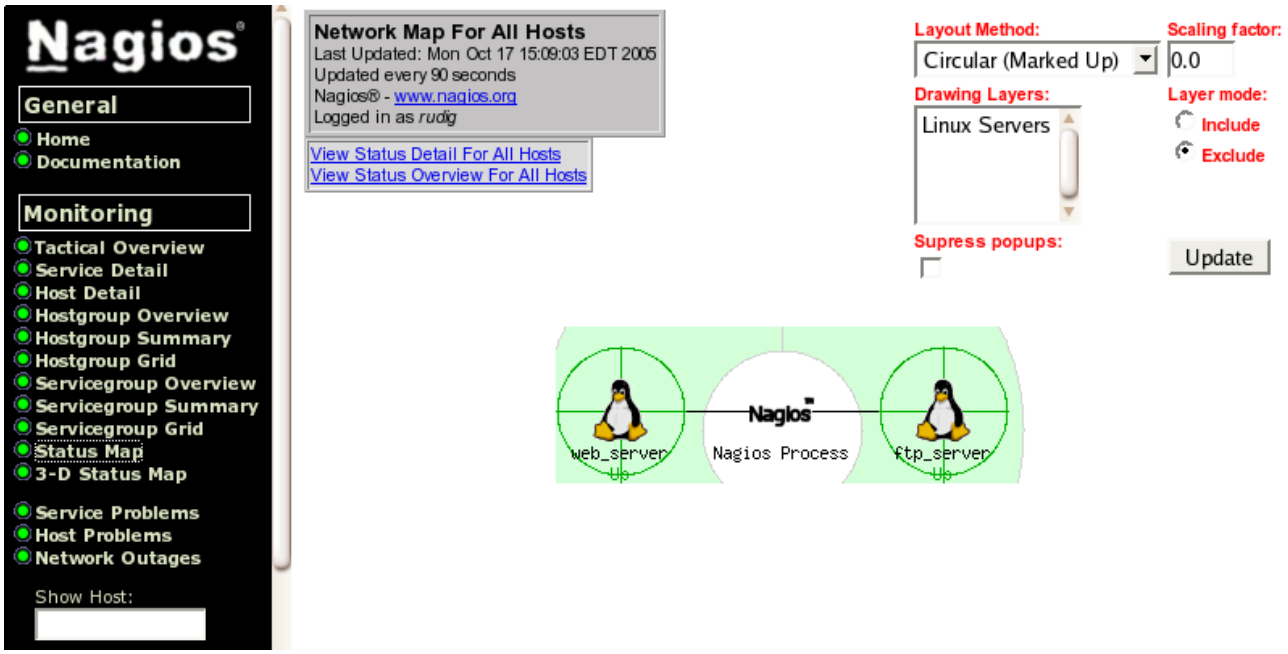
```
define host{
        use                     my_host
        host_name               ftp_server      ;Nome del server
        alias                   FTP server
        address                 192.168.1.137
        }

define hostextinfo {
        host_name               ftp_server
        icon_image              linux.png
        icon_image_alt          FTP
        vrml_image              linux.png
        statusmap_image         linux.gd2
        }

################################################################################
# HOST GROUP DEFINITIONS
################################################################################



# 'linux-boxes' host group definition
define hostgroup{                   ; al gruppo è stato aggiunto il server FTP
        hostgroup_name  linux-boxes
```

```
        alias           Linux Servers
        members         web_server, ftp_server
        }
```

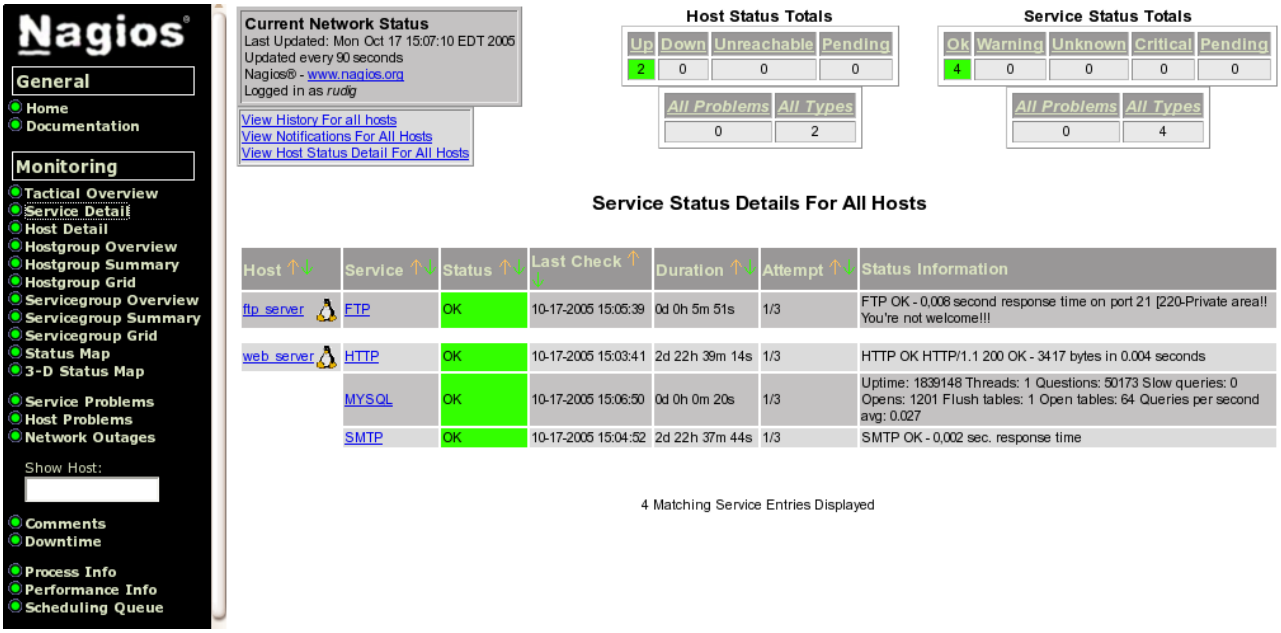
e ora il servizio da controllare...

```
define service{
        use         generic-service         ; Name of service template to use

        host_name                           ftp_server
        service_description                 FTP
        is_volatile                         0
        check_period                        24x7
        max_check_attempts                  3
        normal_check_interval               10
        retry_check_interval                1
        contact_groups                      admins
        notification_interval               120
        notification_period                 24x7
        notification_options                w,u,c,r
        check_command                       check_ftp
        }
```

Dopo il solito test e riavvio otteniamo quanto segue:



Con la corrispondente mappa degli host come da immagine seguente:



La cosa sarebbe banale ma ipotizziamo che dopo circa dieci minuti Nagios ci segnali che entrambi gli host sono interrotti... una rapida verifica e risulta che il problema è lo switch a cui entrambi i server sono collegati. Nagios può tenerne conto!

Configuriamo quindi il sistema in modo che controlli anche lo switch e, in caso di problemi a quest'ultimo ci segnali solo il guasto dell'apparecchiatura e non dei server ad essa collegati. Lo switch in questione ha indirizzo 192.168.1.235

Nel file host oltre ad aggiungere lo switch:

```
# 'switch7' host definition
define host{
        use                     my_host
        host_name               switch7
        alias                   HP switch 7 armadio principale
        address                 192.168.1.235
        }

define hostextinfo {
        host_name               switch7
        icon_image              switch40.png
        icon_image_alt          Switch 7
        vrml_image              switch40.png
        statusmap_image         switch40.gd2
        }
```

Bisogna modificare gli altri host per indicare che dipendono da questo nodo utilizzando il parametro **parents**. Tale informazione fa sì che in caso di problemi di connettività al nodo "padre" non vengano segnalati errori per i nodi figli. A titolo di esempio mostro l'aggiunta per il solo FTP server:

```
# 'ftp_server' host definition
define host{
        use                     my_host          ; Name of host template to use
        host_name               ftp_server
        alias                   FTP server
        address                 192.168.1.137
        parents                 switch7   ; modifica per indicare il nodo padre
```

```
        }
```

Non è necessario monitorare dei servizi sullo switch in quanto il solo PING test dello stesso sarebbe sufficiente a determinare se esso è raggiungibile o meno. Lo switch in questione però è di tipo amministrabile in remoto via SNMP e pur non andando per il momento a prendere in esame questo protocollo, è utile sapere che permette la possibilità di eseguire dei test su tali apparecchi. Andiamo quindi a costruire un comando che verifica il carico della CPU dello switch e configuriamo il servizio relativo.

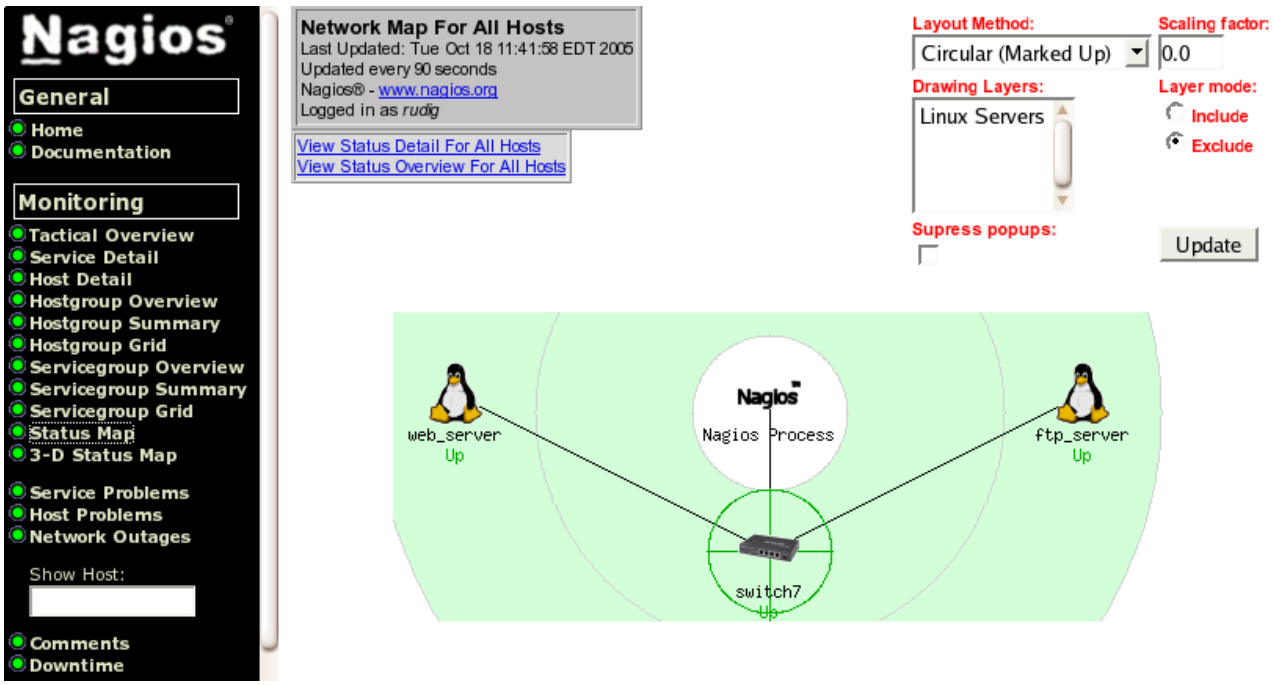
Nel file checkcommands.cfg aggiungiamo:

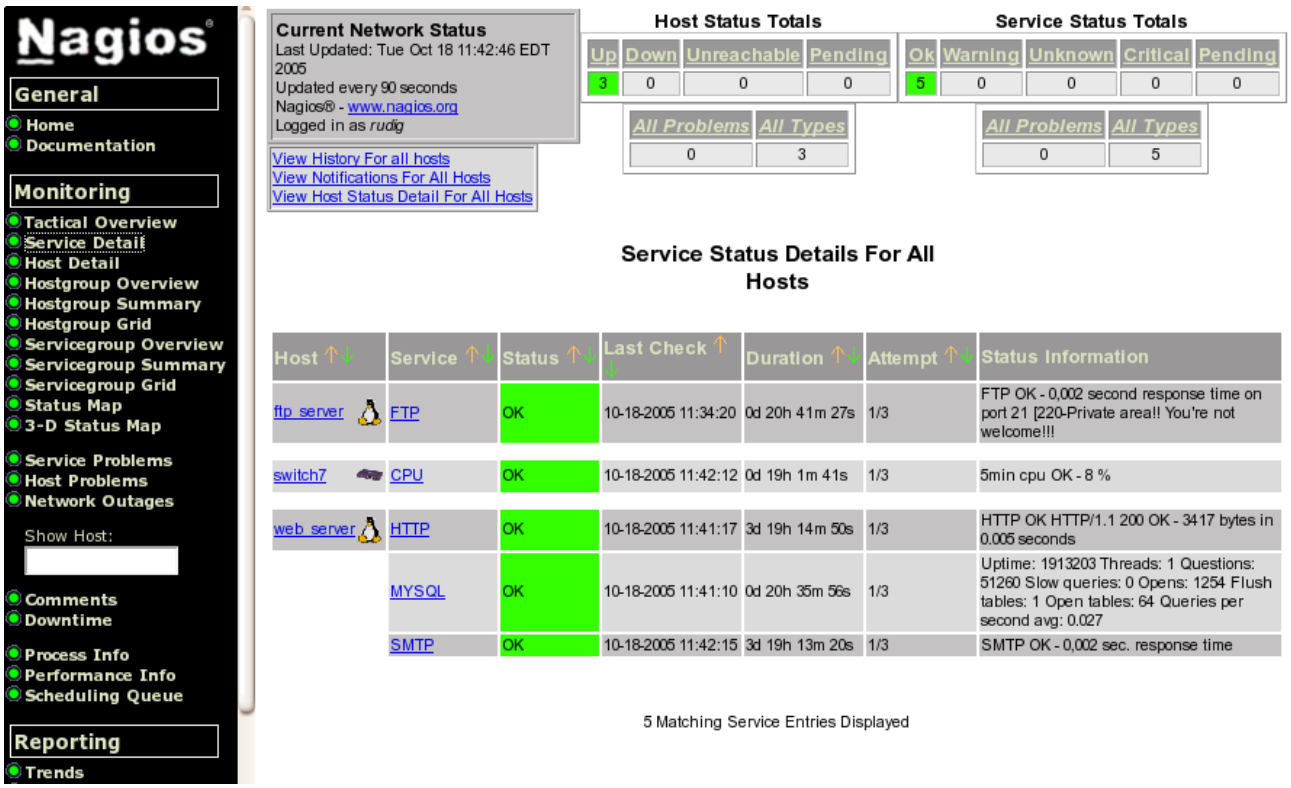
```
define command{
  command_name check_sw_cpu
  command_line $USER1$/check_snmp -H $HOSTADDRESS$ -C $ARG1$ -o .
1.3.6.1.4.1.11.2.14.11.5.1.9.6.1.0 -t 5  -w $ARG2$ -c $ARG3$ -u % -l "5min cpu"
 }
```

e nel file services.cfg

```
define service{
        use                             generic-service
        host_name                       switch7
        service_description             CPU
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              3
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                  admins
        notification_interval           60
        notification_period             24x7
        notification_options            c,r
        check_command                   check_sw_cpu!public!95:90!100:95
 }
```

Ed ecco quanto ci eravamo prefissi.



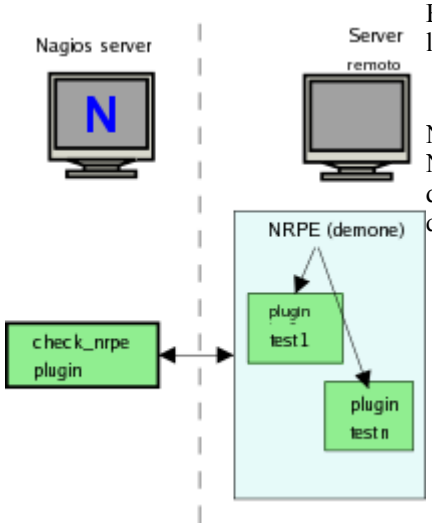


In caso di errore dello switch ci verrà segnalato solo questo e non l'irragiungibilità degli host ad esso collegati. Un effetto collaterale positivo è che abbiamo anche a disposizione il dato relativo al carico della CPU dello switch che può essere utile per individuare un eventuale problema allo stesso.

Controllo indiretto

Oltre ai test diretti, eseguiti tramite i plugins Nagios, mette a disposizione altri due sistemi per eseguire test su host remoti.

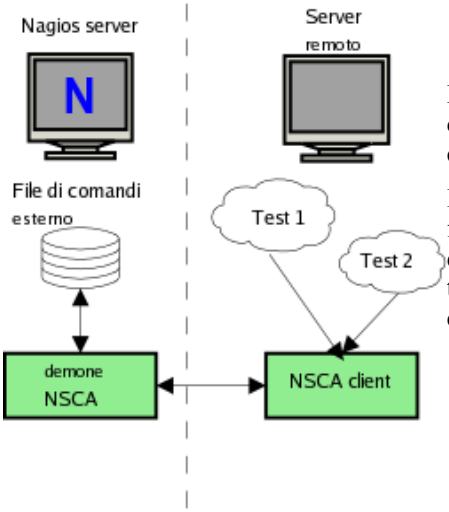
Il primo viene utilizzato quando i servizi che si devono verificare non sono interrogabili in remoto, ad esempio non esiste un modo di verificare in remoto lo spazio libero sul disco di un' altro PC se non di verificarlo localmente al PC. La delega di controllo viene demandata ad un demone chiamato NRPE che viene eseguito sull' host da controllare.



E` un metodo che non ho avuto ancora modo di provare quindi ne espongo solo la teoria come da manuale.

Nagios tramite un plugin di nome *check_nrpe* contatta il demone remoto di NRPE che è in ascolto richiedendo il test. Il demone remoto riceve la richiesta di Nagios esegue i test locali tramite i plugins in modo del tutto analogo a quanto farebbe Nagios ed restituisce la risposta al plugin chiamante.

Trovo molto più interessante parlare dei test indiretti o passivi in quanto permettono di generare qualsiasi tipo di test senza oltretutto pesare sul server Nagios per l'elaborazione.



Per eseguire dei test passivi è necessaria l'installazione sulla macchina remota di un applicativo chiamato NSCA client e, sul server nagios del corrispondente demone NSCA che rimane in ascolto delle comunicazioni.

Il passaggio dei risultati fra il demone NSCA e Nagios avviene attraverso un file definito *external command file* (traducibile come file dei comandi esterno). Uno degli errori più frequenti, che impediscono il corretto uso dei test passivi, è proprio un errata impostazione dei permessi del file in questione.

Vediamo quindi l'installazione del tutto:

decomprimiamo i sorgenti e compiliamo il programma:

```
# tar -zxvf nsca-2.4.tar.tar
# cd nsca-2.4/
# ./configure ; make all
```

Poiché il demone viene eseguito tramite il wrapper tcp ovvero attraverso il demone xinet dobbiamo fare delle modifiche alla configurazione di quest'ultimo: iniziamo con il modificate il file /etc/services aggiungendo la linea seguente:

```
nsca            5667/tcp        # NSCA
```

aggiungete nella directory /etc/xinetd.d un file chiamato nsca che contenga quanto segue:

```
# default: on
# description: NSCA Nagios alert
# version.
service nsca
{
        disable = no
        flags           = REUSE
        socket_type     = stream
        protocol        = tcp
        user            = nagios
        group           = nagios
        wait            = no
        server          = /usr/local/nagios/bin/nsca
        server_args     = -c /usr/local/nagios/etc/nsca.cfg --inetd
        log_on_failure  += USERID
}
```

copiamo i file binario e di configurazione a destinazione:

```
# cp nsca-2.4/src/nsca /usr/local/nagios/bin/
# cp nsca-2.4/nsca.cfg /usr/local/nagios/etc/
```

ed apportate a quest'ultimo le seguenti modifiche:

```
[...]
# ALLOWED HOST ADDRESSES
# nella rete due macchine 192.168.1.6-7 sono autorizzate
# all'invio di messaggi

allowed_hosts=127.0.0.1
allowed_hosts=192.168.1.6
allowed_hosts=172.168.1.7


[...]

# DECRYPTION PASSWORD
# password da inserire sia sul server che sul client

password=mia_pass
```

a questo punto se non esiste bisogna creare il file di comandi esterno, aggiungere il corretto utente, gruppo e gestire i permessi:

```
# touch /usr/local/nagios/var/rw/nagios.cmd
# /usr/sbin/groupadd nagiocmd
# /usr/sbin/usermod -G nagiocmd nagios
# /usr/sbin/usermod -G nagiocmd nobody
# chown nagios.nagiocmd /usr/local/nagios/var/rw
# chmod u+rwx /usr/local/nagios/var/rw
# chmod g+rwx /usr/local/nagios/var/rw
# chmod g+s /usr/local/nagios/var/rw
```

riavviate il servizio con il comando

```
service xinetd restart
```

ed il server NSCA è pronto ad accogliere i messaggi del client.

Il client in caso di diversa architettura va compilato sulla macchina da monitorare. Se avete la stessa versione e distribuzione di GNU/Linux o di UNIX su entrambe le macchine potete semplicemente copiare il file generato nella compilazione precedente.

Per omogeneità di struttura ho creato dei percorsi analoghi a quelli del server Nagios su ciascuno dei due client da monitorare. La serie di comandi che seguono dovrebbe essere esplicativa:

```
$ pwd
/usr/local/nagios
[rudig@venus nagios]$ ls -l
totale 8
drwxr-xr-x    2 root     root         4096 16 mag 15:20 bin
drwxr-xr-x    2 root     root         4096 16 mag 16:35 etc

$ ls bin
send_nsca
[rudig@venus nagios]$ ls etc
send_nsca.cfg
```

Il file send_nsca.cfg è stato modificato dallo standard inserendo la riga relativa alla password:

```
[...]

# DECRYPTION PASSWORD
# password da inserire sia sul server che sul client
```

```
password=mia_pass
```

Tutto ciò rappresenta la pura e semplice fase di installazione. Ora è necessario configurare un servizio da controllare.

Si porterà ad esempio un caso reale: c'era la necessità di controllare il numero di processi di un determinato applicativo su un server. Questo applicativo è costituito da un processo che elabora una lista di transazioni su un server SQL.

Se nessuna istanza dell'applicativo è attiva le transazioni si accumulano in coda e i dati mostrati non sono più aggiornati. Questo non è un errore grave in quanto una volta fatto ripartire questo motore i dati si riallineano ma è opportuno che ci sia una segnalazione un modo da poter riavviare il processo dopo aver verificato il motivo del blocco.

Se invece per un errore vengono avviate più istanze del processo è possibile che la coda venga elaborata fuori sequenza creando degli errori nelle transazioni. Gli errori non sono certi ma possibili. Questa condizione va quindi evitata per quanto possibile.

Si desiderava quindi

- ricevere una segnalazione di corretto funzionamento quando un solo processo dell'applicativo è attivo
- ricevere un Warning (segnalazione di attenzione) in caso di processo assente
- ricevere un avviso di errore critico in caso di istanze multiple del processo

E` stato definito il server da controllare in hosts.cfg ed un servizio in grado di notificare gli stati in questione nel file services.cfg ed il comando check_null in checkcommands.cfg per impostare un test attivo nullo:

```
# 'app_server' host definition
define host{
        use                     my_host
        host_name               app_server          ;nome del server
        alias                   Application server
        address                 192.168.1.6
        }

define hostextinfo {
        host_name               app_server
        icon_image              linux40.png
        icon_image_alt          Linux Host
        vrml_image              linux40.png
        statusmap_image         linux40.gd2
        }


define service{
        host_name                       app_server
        service_description             test_s_process     ;nome arbitrario
        is_volatile                     1           ; sempre a 1 per nsca
        active_checks_enabled           0           ; sempre a 0 per nsca
        check_period                    none
        passive_checks_enabled          1           ; è un servizio passivo
        max_check_attempts              1
        normal_check_interval           10
        retry_check_interval            1
        contact_groups                  admins
        notification_interval           120
        notification_period             24x7
        notification_options            w,u,c,r
        check_command                   check_null
        notifications_enabled           1   ; notifica attiva
        }

# 'check_null' command definition by rg
define command{
        command_name    check_null
        command_line    $USER1$/check_dummy
```

```
        }
```

Poi è stato necessario inventare un metodo per verificare, nel server da controllare, l'attività dell'applicativo in questione tenendo presente che il messaggio che viene trasmesso da send_nsca deve avere il seguente formato:
`<nome dell'host> [tabulazione]<nome del servizio>[tab]<codice di ritorno>[tab]<descrizione><carattere di new line>`

nell'esempio

`app_server  test_s_process    0     Tutto OK`

dove il codice di ritorno è

0 = se tutto funziona

1 = per generare un WARNING

2 = per generare un errore CRITICAL

Nel nostro caso è stato creato lo script seguente che è ampiamente commentato:

```
# cat /usr/bin/sendalert.sh

#!/bin/sh
#sendalert.sh
#Script per il test del motore dell'applicativo

#motore_app è il nome del processo sotto controllo elenco con ps i processi
#filtro con grep quelli che contengono il termine con il nome applicativo e
#conto le linee con wc -l
NUMINST=$(ps ax | grep motore_app | wc -l)
#poichè compare anche la linea precedente nel computo dei processi segnalati da
PS
#è necessario eliminare uno dal conteggio.
NUMINST=$[NUMINST-1]

#in base al numero di processi calcolato inviamo il risultato al server nagios
case $NUMINST in
        0)
                /usr/bin/printf "%s\t%s\t%s\t%s\n" "app_server"
"test_s_process" "1"  "WARNING! Processo applicativo non  attivo" |
/usr/local/nagios/bin/send_nsca -H 192.168.1.4 -c
/usr/local/nagios/etc/send_nsca.cfg


        ;;
        1)
                /usr/bin/printf "%s\t%s\t%s\t%s\n" "app_server"
"test_s_process" "0"  "OK! Processo applicativo attivo" | /usr/local/nagios/bin/
send_nsca -H 192.168.1.4 -c /usr/local/nagios/etc/send_nsca.cfg


        ;;
        2)
                /usr/bin/printf "%s\t%s\t%s\t%s\n" "app_server"
"test_s_process" "2"  "CRITICAL! Troppe istanze Processo applicativo attive" | /
usr/local/nagios/bin/send_nsca -H 192.168.1.4 -c
/usr/local/nagios/etc/send_nsca.cfg
        ;;
esac

exit
```

Ora per effettuare il controllo è sufficiente schedulare nel cron un esecuzione dello script appena preparato con un intervallo di tempo appropriato:

```
# crontab -e
```

aggiungere:

```
## Verifica del processo
05 * * * * /usr/bin/sendalert.sh
```

Ciò che risulta a video nella pagina relativa ai servizi è una riga come la seguente:

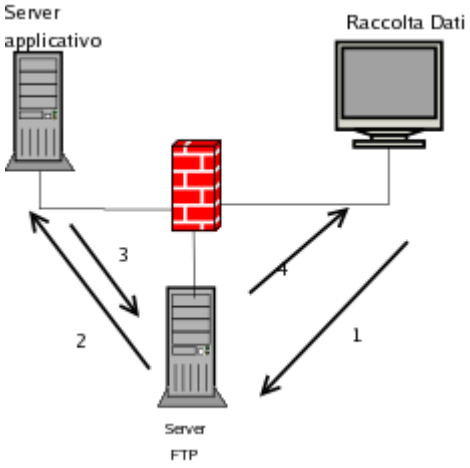
tests process PASSV	OK	10-20-2005 16:27:14	2d 6h 23m 8s	1/1	OK! Processo applicativo attivo

in cui il simbolo dopo il nome del test indica in maniera evidente che il servizio è di tipo passivo.

Controllo di eventi asincroni

Il controllo tramite NSCA è particolarmente indicato per il monitoraggio di eventi asincroni. Vorrei portare ad esempio un caso limite che mostra l'adattabilità di Nagios anche se probabilmente vi sono altri strumenti in grado di effettuare lo stesso tipo di verifiche.
Un modulo dell'applicativo, di cui all'esempio precedente, scambia dei dati con degli applicativi esterni tramite importazione/esportazione di file ASCII. Per il corretto funzionamento di entrambi gli applicativi è necessario che ogni esportazione verso la periferia sia preceduta dalla relativa importazione dei dati precedenti nel sistema centrale. Lo scambio dati avviene attraverso un server FTP.

Una schematizzazione del tutto è la seguente:



La soluzione del problema è apparentemente semplice in quanto sarebbe sufficiente verificare la presenza del file di import nel server FTP prima di procedere all'export. La situazione è complicata dal fatto che, per logiche applicative, il nome del file cambia ogni giorno avendo come prefisso la data. Cambia inoltre anche l'ora di trasmissione in quanto la stessa è a discrezione dell'operatore e può avvenire al mattino o al pomeriggio.

Si è pensato quindi di ribaltare il problema. Si suppone che sia sempre presente un errore di trasmissione a meno di una corretta trasmissione.
Viene quindi usato un file come semaforo:

1) Uno script nella crontable del server applicativo predispone un file di errore ogni mattino.
2) Uno script nel server FTP cancella ogni mattina il file precedentemente inviato dal PC di raccolta dati.
3) Il PC di raccolta dati contiene nella cartella dei dati un file semaforo che contiene l'indicazione di corretto funzionamento dell'importazione dati.

Ipotizziamo di chiamare il servizio test_trasf.
In accordo con la sintassi già vista in precedenza prepareremo nel server applicativo un file test_trasf.sem con il seguente contenuto.

```
app_server   test_trasf 1       Warning! File non spediti da raccolta dati
```

Il file verrà copiato alla mattina in una cartella specificata come ad esempio /temp/import ed indicherà che il trasferimento dati non è andato a buon fine.
Nel PC di raccolta dati è presente un file con lo stesso nome ma contenuto indicante il corretto trasferimento:

```
app_server   test_trasf 0       OK! File spediti da raccolta dati
```

quindi quando l'utente sincronizza i dati questo file viene trasferito nel server FTP.
Il server applicativo ritira i files dal server applicativo portandoli in /temp/import ove il file semaforo sovrascrive quello che indica l'errore.
Durante gli orari non lavorativi e prima dell'esportazione dati serale viene eseguito lo script seguente che invia i dati al server Nagios tramite NSCA.

```
sendtrasf.sh
#!/bin/sh

# per trasf
cat /mnt/intra/trasf.sem | /usr/local/nagios/bin/send_nsca -H 192.168.1.4 -c
/usr/local/nagios/etc/send_nsca.cfg
```

Nel server Nagios al file service.cfg è stato aggiunto il servizio che segue:

```
define service{
        host_name                       app_server
        service_description             test_trasf
        is_volatile                     1
        active_checks_enabled           0
        check_period                    none
        passive_checks_enabled          1
        max_check_attempts              1
        normal_check_interval           10
        retry_check_interval            1
        contact_groups                  admins
        notification_interval           120
        notification_period             24x7
        notification_options            w,u,c,r
        check_command                   check_null
        notifications_enabled           1   ; Service notifications are enabled

        }
```

quindi due volte al giorno viene visualizzato lo stato delle importazioni come segue:

in caso di errore viene inviata una notifica al personale di assistenza che può intervenire per richiedere un nuovo invio di dati o bloccare l'esportazione dei dati ed evitare la sovrascrittura di quelli in ingresso.

Come anticipato vi sono sicuramente altri metodi, forse migliori, per eseguire verifiche di questo tipo ma questo esempio è operativo e funzionante in ambiente reale e si è reso più volte utile per evitare errori.

Ripristino automatico

A partire dalla versione 2.0 Nagios possiede una funzionalità di ripristino automatico di un servizio tramite una funzione detta *"event handling"* ovvero gestione di eventi. Per gestire tale funzione è sufficiente dichiarare all'interno di un servizio un riferimento ad un "event handler". L'esempio che segue è preso pari-pari dal manuale:

```
define service{
        host_name                       somehost
        service_description             HTTP
        max_check_attempts              4
        event_handler                   restart-httpd
        ...other service variables...
        }
```

Come vedete la quarta riga richiama una funzione "restart-httpd" che ovviamente serve per riavviare il demone httpd nel caso risultasse indisponibile.
Tale funzione viene definita all'interno di uno dei files di configurazione come ad esempio checkcommands.cfg e conterrà il nome della funzione o del comando atto a gestire il servizio in questione come ad esempio:

```
define command{
        command_name    restart-httpd
        command_line    /usr/local/nagios/libexec/eventhandlers/restart-httpd
$SERVICESTATE$ $SERVICESTATETYPE$ $SERVICEATTEMPT$
        }
```

qui si vede viene richiamata una command_line che come espone il manuale è un comando creato per l'occasione e che non funziona.
Per meglio chiarire: sicuramente non funziona in nessuna delle distribuzioni Red-Hat o Fedora in cui ho avuto modo di provarlo e concettualmente ciò è corretto. Il fatto è che per ragioni di sicurezza ci siamo dati tanto da fare affinché il

demone Nagios e tutti i processi dipendenti avessero scarsi privilegi e permessi in modo da evitare una escalation in caso di intrusioni. Il fatto è che gestire e riavviare i processi Nagios deve necessariamente avere dei privilegi più elevati altrimenti non ha permesso di accesso ai file semaforo dei servizi e a volte nemmeno agli script che gestiscono i servizi stessi.
Non ci sono rimedi ovvi: O si abbassano i livelli di sicurezza o si rinuncia a questa funzionalità. La scelta dipende dalla necessità di continuità del servizio rapportata con i requisiti di sicurezza. Per quel che mi riguarda ho preferito mantenere elevati gli standard di sicurezza ed ho evitato di proseguire i test su questa funzionalità

-- o --

Poiché molti dei test di Nagios possono essere eseguiti via SNMP vediamo ora di esaminare un po questo protocollo e le opportunità offerte.

SNMP

Il *Simple Network Management Protocol* è un protocollo per l'amministrazione delle reti definito dallo standard internet RFC1157 [http://www.faqs.org/rfcs/rfc1157.html] integrato da numerosi altri RFC successivi. Si tratta quindi di un protocollo e non di un applicazione specifica e fu sviluppato per fornire risposte alla necessità di un protocollo di amministrazione per le reti con gli obiettivi principali di semplicità e robustezza. Nelle specifiche veniva previsto un basso sovraccarico sia per i dispositivi che utilizzassero il protocollo sia per la rete.

La prima volta che ho *affrontato* questo protocollo ho fatto l'errore, comune a molti, di tradurre *simple* con *semplice* (nel senso di facile da usare) mentre una traduzione più consona potrebbe essere *semplificato* nel senso che la struttura e i metodi del protocollo sono semplificati. Vengono infatti supportati solo tre metodi fondamentali:

- la scrittura di una variabile – comando **set**
- la lettura di una variabile – comando g**et**
- la notifica di un evento - **trap**

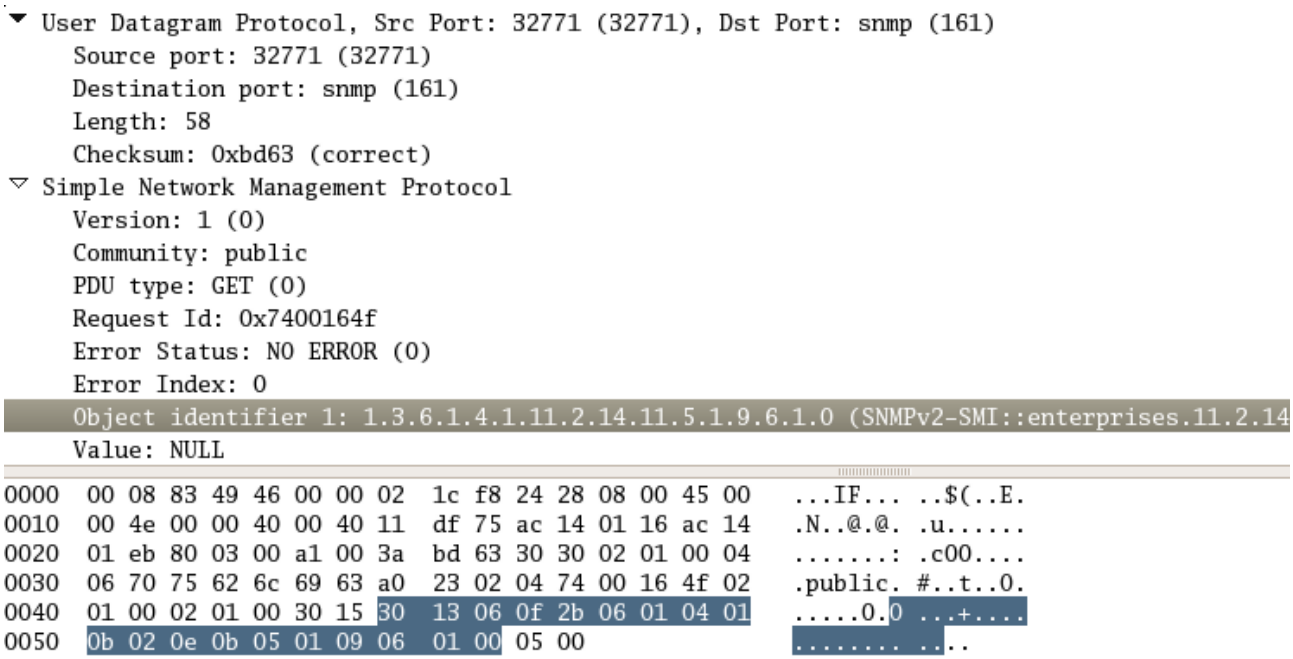
Non si desidera di seguito entrare troppo nel dettaglio del protocollo ma è importante evidenziare che la semplicità è proprio nella struttura del pacchetto che consiste sempre di un pacchetto UDP monolitico (l'aggettivo inglese *atomic* rende meglio se tradotto con monolitico a mio parere).
Il pacchetto è costituito da:
- un identificativo di versione,
- un *community name (*che praticamente rappresenta una password),
- un unità dati (Protocol Data Unit).

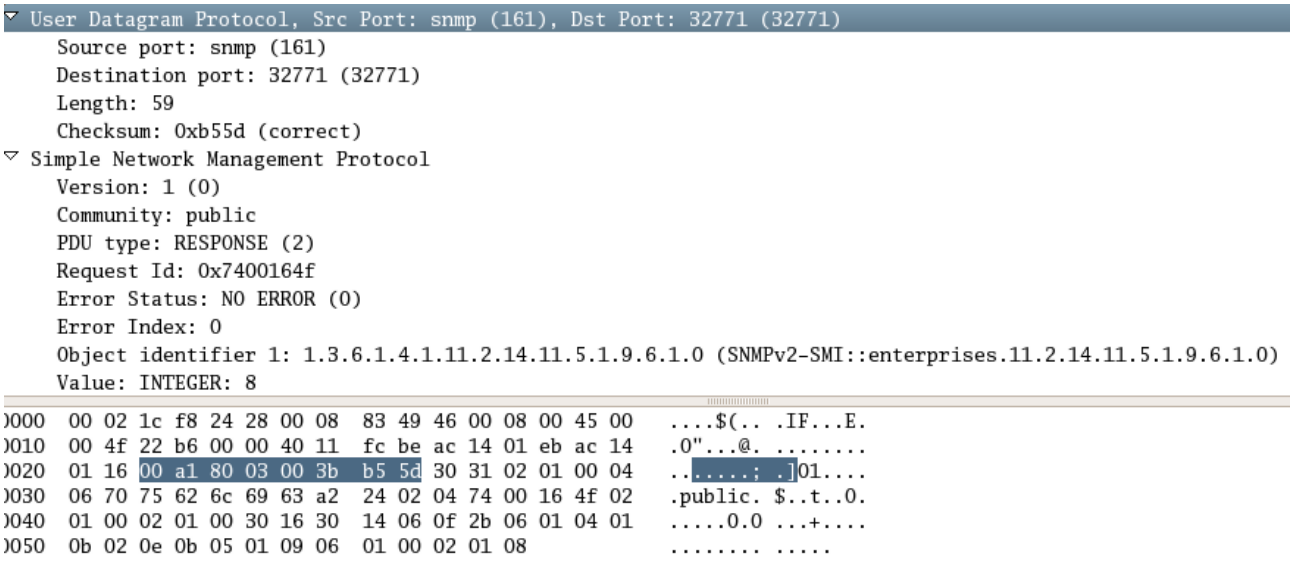
Le PDU contengono sia un identificativo di tipo sia i dati/variabili su cui si va ad agire.
Ci sono solo cinque tipi di PDU che sostanzialmente corrispondono a cinque diverse azioni: **get-request, get-next-request, set-request, get-response** e **trap**.

Un pacchetto get-request, inviato a un dispositivo, ottiene in risposta un get-response che restituisce il valore di una variabile.
Un pacchetto get-next-request è utilizzato per iterare fra gli elementi di un albero di variabili e ottiene sempre un get-response.
Un pacchetto set-request configura una variabile di un dispositivo. Anch'esso ottiene in risposta un get-response.

I trap sono differenti in quanto vengono inviati dal dispositivo ad un indirizzo IP configurato nel dispositivo stesso al verificarsi di determinati eventi per i quali il dispositivo è programmato. Il pacchetto in questione contiene una serie di variabili/valori che il dispositivo ritiene utili per descrivere l'evento. Non viene attesa risposta al trap. Cosa succede all'indirizzo di destinazione o, perfino, se esista qualcosa a quell'indirizzo non è a carico del dispositivo il quale si limita all'invio della segnalazione.
Le porte usate dal protocollo sono la UDP 161 per le richieste e le risposte e la UDP 162 come destinazione delle trap SNMP; è necessario tenerne conto quando gli apparecchi da monitorare e quelli di monitoraggio sono posti sui due lati di un firewall.
La versione 2, e successive, del protocollo implementano anche dei comandi aggiuntivi e delle estensioni agli oggetti gestiti. Vi rimando alla lettura degli RFC 1441 e 1442 per ulteriori dettagli.

Nelle immagini che seguono viene mostrato un pacchetto UDP inviato per richiedere un valore ad uno switch e la successiva risposta. Il pacchetto è stato rilevato con ethereal [http://www.ethereal.com/] e come si vede si tratta una richiesta di una variabile (GET) e della successiva risposta. In entrambe le immagini è identificabile la community string di default ("public").





Come già detto SNMP offre, per ogni dispositivo, una variabile detta *comunity string* che ha la funzione di una password. Si può vedere dalle immagini che la stringa in questione è visibile in chiaro, quindi, in realtà è una sicurezza debolissima ed è uno dei motivi per i quali sono state, teoricamente, abbandonate le versioni 1 e 2c del protocollo (anche se in realtà sono ancora le più utilizzate) a favore della versione 3.

Come già anticipato SNMP tratta degli oggetti che essenzialmente sono delle variabili. Tali variabili sono definite e descritte nella Management Information Base (MIB). MIB non è un database ma una struttura dati ovvero un file scritto in uno specifico linguaggio (SMI – Structure of Management Information) che elenca delle variabili assegnando ad ogni variabile un nome, un numero ed un elenco di permessi.

Il file viene visto come una gerarchia ad albero e ogni variabile è considerata una foglia nell'albero. Tutto ciò che

riguarda SNMP si trova al di sotto del ramo denominato **.iso.org.dod.internet** che ha come identificativo numerico **1.3.6.1**

L'immagine seguente è uno snapshot dal sito http://support.ipmonitor.com/ e rappresenta un esempio di gerarchia ad albero

L'identificativo numerico di ciascun oggetto è detto OID (Object Identifier) e non dovrebbe iniziare con un punto anche se, in molte notazioni, il punto iniziale viene sempre riportato. E' utile tenere presente che, in una qualsiasi operazione, qualora l' OID specificato non sia completo (ovvero qualora venga specificato un nodo anziché un oggetto) tutti gli oggetti del nodo richiesto saranno forniti in risposta.
Oltre all'albero standard definito dal comitato di standardizzazione esistono dei sotto-alberi registrati da aziende private presso l'ente di standardizzazione. Per impieghi interni all'azienda è, ovviamente, possibile definire un proprio sotto-albero qualora ve ne fosse la necessità.

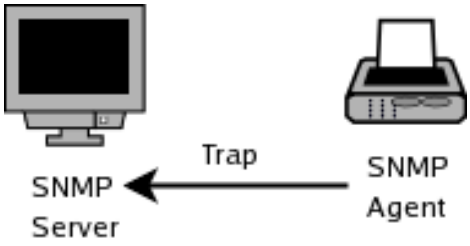
Gli oggetti veri e propri, o variabili che dir si voglia, possono essere di cinque tipi:

- Stringhe di caratteri – solitamente rappresentano delle descrizioni di altre variabili o nomi descrittivi o frasi che compaiono nei display.
- Octect String (otteti) può rappresentare una stringa di caratteri o un dato binario secondo quanto specificato dal MIB
- Interi – usati solitamente come indici per tabelle
- Contatori – interi che crescono continuamente fino al loro limite e poi si azzerano. Sono a 32 o 64 bit
- Gauge (traducibile con strumento di misura/indicatore) - è una variabile che può aumentare o diminuire nel tempo ed è utilizzata per misurare un valore es. carico di una CPU, velocità di una ventola o bps di un interfaccia.

In base al tipo di dispositivo vi sono un infinità di parametri che possono essere misurati o rilevati con SNMP ad esempio il carico di una CPU, il numero di processi di un server, i pacchetti in errore di un router, le collisioni di una rete, la memoria libera di una stampante e così via.

Il dispositivo che si incarica del monitoraggio SNMP e/o della ricezione dei trap, sia esso un server, un PC o un tester dedicato viene denominato Network Management Station (NMS).

Nel dispositivo da monitorare o amministrare, il protocollo SNMP, prevede la presenza di un programma, detto agente. Vi sono agenti SNMP nei routers, negli switch, nelle stampanti, ad esempio, ma vi sono agenti disponibili anche per i sistemi operativi sia per i vari UNIX e derivati sia per le varie versioni di Windows. L'agente oltre a rispondere a delle richieste dirette di tipo lettura o modifica di una variabile potrebbe venir programmato (qualora sia programmabile) per generare un trap in risposta ad un determinato evento.



Un esempio semplice è quello delle stampanti; molte di esse hanno il protocollo SNMP attivo al loro interno e sono in grado di generare un avviso per eventi quali l'esaurimento della carta o del toner. Il segnale in questione può essere monitorato da un utility dedicata o da un NMS predisposto per ricevere la notifica del problema.

Il vantaggio delle trap rispetto all'interrogazione delle variabili è che con quest'ultimo metodo può accadere di perdere un evento fra un interrogazione e l'altra mentre nel primo caso è proprio l'evento a creare la notifica e quindi è meno probabile che esso venga perso. La perdita di una segnalazione è comunque sempre possibile in quanto potrebbe accadere che, il server che gestisce le notifiche, se ha in carico un numero troppo elevato di dispositivi e molti di essi sono in errore, può non essere in grado di gestire tutte le segnalazioni. Chiaramente la situazione è improbabile in un ambiente ben dimensionato. Inoltre un server che interroga attivamente le variabili è soggetto ad un maggior carico elaborativo rispetto ad uno che si limita ad attendere passivamente la segnalazione di un problema e quindi l'interrogazione attiva sarebbe comunque meno efficiente.

Chiaramente se non si deve monitorare un evento ma effettuare delle misure a intervalli prestabiliti il metodo corretto è il polling da parte della NMS e non l'utilizzo delle trap.
Per quanto riguarda il carico creato dal polling sulla rete riporto una divertente indicazione di John Blommers:
"State per implementare un sistema di monitoraggio delle performance della rete basato su SNMP:

- *la misurazione dei pacchetti di get e relativa risposta danno una cifra fra i 200 e i 250 bytes*
- *250 è il valore più alto, utilizziamolo per i calcoli*
- *si vuole mantenere il traffico SNMP al di sotto del 10% del link più lento della WAN*
- *per ciascuna rete contate il numero di misure da rilevare*
- *assumete di utilizzare un intervallo di 1 minuto come punto di partenza*
- *determinate i percorsi dei pacchetti SNMP attraverso l'intera rete*
- *aggiungete i flussi che utilizzano lo stesso percorso per i pacchetti con dimensione maggiore di 250 byte*
- *confrontate il flusso totale con la velocità della linea in ciascun punto*
- *per flussi maggiori del 10% previsto riducete l'intervallo di polling o il numero di dispositivi monitorati.*
  Poi chiedetevi: perché gli amministratori di rete si devono preoccupare per l'aggiunta del traffico relativo al monitoraggio quando web e mail server vengono installati senza alcun dimensionamento e gli utenti navigano in rete senza riguardo o preoccupazione su quale impatto ciò abbia sulla rete?"

RMON

RMON è un MIB introdotto dalla versione 2 di SNMP che definisce uno standard per il monitoraggio di rete. In tale mib che inizia da **.iso.org.dod.internet.mgmt.mib-2.rmon,** e che ha come identificativo numerico **1.3.6.1 2.1.16** , sono definite 9 sezioni:

1. statistic: mantiene dati relativi agli errori e all'utilizzo per ogni sotto-rete monitorata bytes, pacchetti, collisioni etc.
2. history: campionamenti periodici prelevati tipicamente ogni 30 secondi
3. alarm: gruppo che permette la definizione di valori di soglia, da utilizzare per gli alarmi, per ogni variabile di tipo contatore
4. host: contatori per ogni host della sotto-rete
5. hostTopN: statistiche relative agli host
6. matrix: matrice di errori e utilizzo
7. filter: si possono definire dei filtri da utilizzare per catturare i pacchetti che corrispondono al filtro definito
8. capture: indicazioni sulla modalità di invio dei dati alla console di monitoraggio
9. event: tabella di eventi generati dall'agente RMON

Chiaramente a seconda del dispositivo ciascun gruppo può essere o non essere implementato e avere o non avere un senso.

Ora, dopo aver ribadito più volte che SNMP è un protocollo e dopo aver capito che disponiamo di un database di informazioni (MIB) vediamo un implementazione che utilizza tale protocollo e passiamo dalla teoria alla pratica.

Net SNMP

Il progetto Net SNMP, precedentemente noto come UCD-SNMP, è una raccolta di strumenti per gestire le informazioni SNMP in ambiente *NIX. Attraverso vari programmi permette le varie operazioni di lettura, scrittura e monitoraggio del protocollo fornendo sia quanto necessario per l'implementazione di un client sia gli strumenti per la gestione di una NMS. Il sito ufficiale è http://www.net-snmp.org/

I MIB supportati dal pacchetto sono:

- MIB-2: Statistiche di rete conforme a RFC1213
- Risorse degli host come da RFC1514 e 2790
- SNMPv3 MIBS con il supporto alla versione 3
- MTA-MIB con il supporto a sendmail
- Estensioni a MIB privati

La prima versione di Net SNMP (la 5.0 in quanto le versioni precedenti sono state rilasciate come UCD-SNMP) ha adottato dall'inizio dei concetti di modularità, che non erano presenti in nella versione UCD, per cui, a titolo di esempio, per aggiungere un MIB a quelli in dotazione e sufficiente copiarlo in /usr/local/share/snmp/mibs (o /usr/share/snmp/mibs a seconda della distribuzione e del sistema operativo in uso) e con la stessa relativa semplicità è possibile estendere gli applicativi.

Non andrò, di seguito, a spiegare l'installazione dei programmi visto che ne esiste praticamente un pacchetto per ogni distribuzione, posso solo segnalare che nella versione per Fedora manca una qualche forma di collegamento fra il pacchetto del modulo SNMP perl e quello di Net SNMP e il browser grafico dei mib (tkmib) non funziona.

Non andrò nemmeno a spiegare in dettaglio tutti i comandi in quanto le man-pages sono esaustive in merito, ma mi limiterò ad una carrellata sui comandi principali con qualche esempio d'uso.

Snmpwalk:

Il primo comando da prendere in considerazione è **snmpwalk**. Questo comando produce come risultato la stampa dell'intero albero delle variabili di un dispositivo a partire dal punto passato come parametro. Se non si fornisce alcun punto di partenza il risultato sarà l'intero albero.

Ecco un esempio di output di snmpwalk che si riferisce ad uno switch:

```
# snmpwalk -m all 192.168.1.254 -c public
system.sysDescr.0 = "HP J4813A ProCurve Switch 2524, revision F.02.11, ROM
F.02.01  (/sw/code/build/info(f00))"
system.sysObjectID.0 = OID: enterprises.11.2.3.7.11.19
system.sysUpTime.0 = Timeticks: (244993657) 28 days, 8:32:16.57
system.sysContact.0 = ""
system.sysName.0 = "Switch 5"
system.sysLocation.0 = "Corridoio 1"
system.sysServices.0 = 74
interfaces.ifNumber.0 = 28
interfaces.ifTable.ifEntry.ifIndex.1 = 1
interfaces.ifTable.ifEntry.ifIndex.2 = 2
interfaces.ifTable.ifEntry.ifIndex.3 = 3
interfaces.ifTable.ifEntry.ifIndex.4 = 4
interfaces.ifTable.ifEntry.ifIndex.5 = 5
interfaces.ifTable.ifEntry.ifIndex.6 = 6
interfaces.ifTable.ifEntry.ifIndex.7 = 7
interfaces.ifTable.ifEntry.ifIndex.8 = 8
[...]
```

il resto dell'output è stato troncato per brevità in quanto continuerebbe per molte pagine. A tal proposito, quando dovete esplorare un MIB, è molto utile redirigere l'output del comando verso un file in quanto spesso il buffer video non riesce a contenerlo completamente.

Indicando un solo ramo (ad esempio *system*) otteniamo un risultato più breve in quanto viene mostrato solo quanto richiesto:

```
# snmpwalk -mall -cpublic 192.168.1.254 system
system.sysDescr.0 = "HP J4813A ProCurve Switch 2524, revision F.02.11, ROM
F.02.01  (/sw/code/build/info(f00))"
system.sysObjectID.0 = OID:
enterprises.hp.nm.system.netElement.hpEtherSwitch.hpSwitchJ4813A
system.sysUpTime.0 = Timeticks: (262307843) 30 days, 8:37:58.43
system.sysContact.0 = ""
system.sysName.0 = "Switch 5"
system.sysLocation.0 = "Corridoio 1"
system.sysServices.0 = 74
```

Questo comando è molto utile in particolare per esplorare il "contenuto" di dispositivi dei quali non si conoscono le variabili. L'assenza di descrizioni in alcuni OID è sintomo della mancanza di un MIB specifico per il dispositivo esaminato e si risolve facilmente, se si riesce a rintracciare il MIB, installandolo come sopra accennato.

E' possibile utilizzare il comando in maniera molto "sporca" per dei monitoraggi improvvisati e a scopo di test. Nel esempio seguente, mancando il MIB specifico per una stampante di rete si voleva verificare se un certo ramo individuato era proprio quello relativo allo stato stampante.
Per individuare il ramo prima è stato eseguito un comando snmpwalk su tutto l'albero poi, per affinare la ricerca, è stata tolta la carta da due cassetti ed è stato eseguito il comando che segue:

```
# snmpwalk -mall -cpublic 192.168.1.236 43.18.1.1.8.1
43.18.1.1.8.1.3 = "Carta assente: Cassetto 4 {13500}"
43.18.1.1.8.1.84= "Carta assente: Cassetto 1 {13200}"
```

A questo punto uno script con un ciclo infinito ha permesso di verificare che il ramo individuato era quello corretto:

```
# vi testerrori.sh
```

inseriamo i seguenti comandi:

```
#!/bin/sh
while [ 1 ]; do
  snmpwalk -mall -cpublic 192.168.1.236 43.18.1.1.8.1
done
#Per bloccare lo script premere ctrl-C
```

e rendiamo lo script eseguibile:

```
# chmod 777 testerrori.sh
# ./testerrori.sh

43.18.1.1.8.1.90 = "Non rilevato: Cassetto 2 {12301}"
43.18.1.1.8.1.90 = "Non rilevato: Cassetto 2 {12301}"
43.18.1.1.8.1.90 = "Non rilevato: Cassetto 2 {12301}"
43.18.1.1.8.1.90 = "Non rilevato: Cassetto 2 {12301}"
43.18.1.1.8.1.90 = "Non rilevato: Cassetto 2 {12301}"
43.18.1.1.8.1.91 = "Non rilevato: Cassetto 3 {12401}"
43.18.1.1.8.1.91 = "Non rilevato: Cassetto 3 {12401}"
43.18.1.1.8.1.91 = "Non rilevato: Cassetto 3 {12401}"
43.18.1.1.8.1.91 = "Non rilevato: Cassetto 3 {12401}"
43.18.1.1.8.1.91 = "Non rilevato: Cassetto 3 {12401}"
```

Il risultato è stato ottenuto estraendo i cassetti. Ovviamente il monitoraggio vero è proprio è stato effettuato con altri strumenti ma niente vieta di migliorare lo script di cui sopra per gestioni molto semplici.

snmpget:

Il comando snmpget permette di ottenere il valore di una singola variabile: per esempio:

```
snmpget -Cf -mall 192.168.1.254 -c public .1.3.6.1.4.1.11.2.14.11.5.1.9.6.1.0
enterprises.11.2.14.11.5.1.9.6.1.0 = 13
```

Come si vede la forma dell' OID nella risposta è abbreviata e manca tutto ciò che precede *enterprises.* La parte finale è invece in forma numerica per mancanza dello specifico MIB relativo al dispositivo interrogato.

Una forma analoga con un MIB più completo è:

```
snmpget 192.168.1.236 public system.sysDescr.0
system.sysDescr.0 = NRG 3525/3508/3502 5.21 / NRG Network Printer C model / NRG
Network Scanner C model
```

che, in questo esempio, permette di ottenere la descrizione del sistema (si tratta di una stampante).

Rifacendosi all'esempio citato nel comando snmpwalk, è possibile controllare la presenza della carta nella stessa stampante, in maniera molto più efficiente, con il seguente comando:

```
# snmpget -mall -cpublic 192.168.1.236 43.18.1.1.8.1.2
43.= "Carta assente: Cassetto 3 {13400}"
```

come è facile intuire è semplice costruire degli script che sfruttano il comando smpget e verificando le risposte ricevute segnalano un problema all'amministratore del dispositivo sotto controllo. Il linguaggio solitamente usato a tale scopo è il PERL (vedi ad es. The Cuddletech Guide to SNMP Programming all'indirizzo [http://www.cuddletech.com/articles/snmp/snmp_paper.html]) ma nulla vieta di implementare dei semplici script di shell.

snmptable

A volte alcune informazioni di un mib sono correlate tra di loro come elementi di una tabella. La normale rappresentazione ad albero fa perdere la visione di insieme su tali elementi in quanto è difficile esaminare tali informazioni visualizzandole in maniera sequenziale. Il comando snmptable permette di ottenere una rappresentazione tabellare delle informazioni con l'unico inconveniente di risultare di difficile lettura per tabelle molto ampie.
Tale problema è poi risolvibile limitando l'output del comando a n. caratteri o con un copia ed incolla su un altro programa.
Vediamo come semplice esempio una tabella di indirizzi di uno switch:

```
# snmptable 192.168.1.253 ipAddrTable -cpublic
SNMP table: ip.ipAddrTable


  ipAdEntAddr ipAdEntIfIndex ipAdEntNetMask ipAdEntBcastAddr ipAdEntReasmMaxSize
     127.0.0.1          4124      255.0.0.0                1               65535
 192.168.1.253            29    255.255.0.0                1               65535
```

tale forma è sicuramente più leggibile dell'equivalente visualizzazione ad albero:

```
# snmpwalk 192.168.1.253 ipAddrTable -cpublic
ip.ipAddrTable.ipAddrEntry.ipAdEntAddr.127.0.0.1 = IpAddress: 127.0.0.1
ip.ipAddrTable.ipAddrEntry.ipAdEntAddr. 192.168.1.253 = IpAddress: 192.168.1.253
ip.ipAddrTable.ipAddrEntry.ipAdEntIfIndex.127.0.0.1 = 4124
ip.ipAddrTable.ipAddrEntry.ipAdEntIfIndex. 192.168.1.253 = 29
ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask.127.0.0.1 = IpAddress: 255.0.0.0
ip.ipAddrTable.ipAddrEntry.ipAdEntNetMask. 192.168.1.253 = IpAddress:
255.255.0.0
ip.ipAddrTable.ipAddrEntry.ipAdEntBcastAddr.127.0.0.1 = 1
ip.ipAddrTable.ipAddrEntry.ipAdEntBcastAddr. 192.168.1.253  = 1
ip.ipAddrTable.ipAddrEntry.ipAdEntReasmMaxSize.127.0.0.1 = 65535
```

```
ip.ipAddrTable.ipAddrEntry.ipAdEntReasmMaxSize. 192.168.1.253  = 65535
```

Traps

Come già anticipato una trap è un metodo per inviare da un dispositivo la notifica di un evento ad un NMS. Net SNMP oltre a permettere di creare delle trap ha in dotazione un demone che permette di costruire una NMS. Il demone in questione è **snmptrapd** ed ha la possibilità di ricevere delle trap ed attivare un programma in risposta a tale evento. Di default snmptrapd ascolta la porta UDP 162 per cui se attivate tale demone è opportuno verificare che tale porta non sia bloccata da un firewall.

Per poter intercettare un evento è necessario definire una traphandle (ovvero un aggancio) per tale evento nel file snmtrapd.conf. La sintassi per definire un traphandle è la seguente:

```
traphandle  OID   commando
```

quindi ad esempio per intercettare un errore da una stampante la configurazione potrebbe essere:

```
traphandle  .1.3.6.1.2.1.25.3.2.1.5.1     /usr/local/bin/sendmessage_to_admin.sh
```

dove l'ipotetico comando /usr/local/bin/sendmessage_to_admin.sh è una shell che invia una mail ad un responsabile dell'operatività della stampante. E' possibile passare a tale programma dei parametri predefiniti alcuni dei quali molto utili come ad esempio HOSTNAME e IPADDRESS che rappresentano il nome e l'indirizzo IP dell'host che ha generato la trap

RRDtool

Round Robin Database tool [http://oss.oetiker.ch/rrdtool/] è un programma open source ideato da Tobias Oetiker che permette di memorizzare misurazioni effettuate nel tempo e ricavarne diagrammi. Si basa sul concetto del round robin una tecnica che utilizza un numero finito di elementi e un puntatore all'elemento corrente. I nuovi elementi vengono aggiunti sovrascrivendo i dati più vecchi. In pratica il database è circolare, una volta raggiunta la fine il puntatore si sposta di nuovo sul primo elemento e inizia a sovrascrivere i dati.

I vantaggi di questa tecnica risiedono proprio nel fatto che essendo noto e predeterminato il numero di elementi che compongono il database le sue dimensioni sono fisse, cosa che sgrava l'amministratore da tutti i problemi di manutenzione relativi alla crescita del database.

Un'altra caratteristica di RRDtool è che i valori non vengono memorizzati quando disponibili ma a intervalli di tempo predeterminati. Se durante l'intervallo di raccolta il dato non è disponibile viene memorizzato al suo posto il valore UNKNOWN (sconosciuto) per quell'intervallo. E' chiaro che un alto numero di valori sconosciuti altera i risultati per cui è molto importante assicurare un flusso costante di dati per l'aggiornamento del database.

Un RRD (Round Robin Database) può contenere qualsiasi tipo di dati numerici, non necessariamente interi, con l'unico limite dato dall'applicabilità della sua struttura circolare.

Il timestamp, ovvero la marcatura temporale, del momento della rilevazione del dato è sempre espressa in numero di secondi trascorsi dal 01/01/1970 (time-epoch) ovvero dalla data convenzionale di creazione di Unix.

RRDtool può essere utilizzato per monitorare qualsiasi tipo di dato sia possibile raccogliere in maniera automatica, ma viene soprattutto utilizzato in congiunzione con il protocollo SNMP.

I sorgenti del programma si possono scaricare da http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/pub/ assieme ai wrappers per vari linguaggi. Cercando su internet potrete sicuramente trovare i binari per le maggiori distribuzioni senza molta fatica.

Le librerie richieste sono libart_lgpl , libpng , zlib, freetype, cgilib coerenti con la versione dei sorgenti scaricata. Potrete comunque trovarle allo stesso link dei sorgenti nella cartella lib/.

Prima di introdurre ulteriore teoria vorrei passare a qualcosa di pratico. Vediamo quindi un esempio di utilizzo

parzialmente ripreso da un articolo su O3 Magazine n.4 [http://www.o3magazine.co.uk/o3/issue4/o3-i4-72.pdf]. Lo scopo di questa implementazione è di rilevare e tracciare il carico medio del processore di un personal PC.

Creiamo il database **loadav.rrd** nella directory corrente:

[root@jupiter root]# rrdtool create loadav.rrd --step 10 DS:load:GAUGE:30:0:100 RRA:AVERAGE:0.5:1:9600 \
RRA:AVERAGE:0.5:4:9600 RRA:AVERAGE:0.5:24:6000

da cui si ottiene il file:

[root@jupiter root]# ls -l

-rw-r--r-- 1 root root 202524 10 mag 16:13 loadav.rrd

Per capire meglio è necessario definire alcuni dei parametri anche se ritengo opportuno invitarvi alla lettura della man page per i dettagli:

Il parametro --step: indica che il database dovrà essere aggiornato ogni x (10 in questo caso) secondi; ovvero rappresenta la risoluzione minima delle letture.

DS: è la variabile di riferimento (data source) in questo caso sarà denominata **load** ed essendo essa di tipo GAUGE non verrà memorizzato il cambiamento dall'ultimo valore rilevato ma il valore assoluto del valore rilevato. Altri tipi di variabile sono COUNTER consistente di un contatore ad incremento continuo di cui viene immagazzinato il valore per differenza rispetto all'ultima lettura, DERIVE per un contatore decrescente, ABSOLUTE lavora come counter ma immagazina il valore del contatore e non la differenza. Si possono creare più variabili in contemporanea dichiarando più DS per uno stesso archivio.

Il programma attenderà al massimo 30 secondi (15 di attesa effettiva + 15 di tolleranza) per il valore prima di registrare un valore “UNKNOWN”. Tale attesa è detta heartbeat (letteramente battito cardiaco). Questo è un valore molto delicato in quanto un intervallo lungo significa accettare la possibile perdita di valori intermedi significativi, un intervallo troppo breve significa rischiare di sovraccaricare il sistema e quindi alterare le misurazioni. Il valore è quindi fortemente legato alla natura del dato da misurare.

La variabile può assumere valori compresi fra min 0 e max 100. I valori al di fuori di tale range vengono scartati automaticamente dal sistema in quanto abbiamo imposto che si tratta di valori errati.

Le variabili successive sono riferite ai Round Robin Archives (RRA) cioè sono specifiche relative al dato archiviato. Il termine AVERAGE è riferito alla funzione di consolidamento e significa “MEDIA” ovvero i dati verranno consolidati con un valore medio (nell'esempio in 3 archivi differenti).

Il primo valore, 0.5, indica che al massimo il 50% dei dati può essere di tipo UNKNOWN. Tale valore che di default è 0 è poco utile quando si riescono a fare misure precise ma mostra il suo senso quanto le rilevazioni dei dati sono molto disturbate.

Nel primo RRA viene indicato che ogni lettura sarà memorizzata fino a 9600 letture poiché ci si attende una lettura entro 15 secondi si crea uno storico di (15 secondi * 9600 letture) = 144.000 secondi memorizzati = 40 ore.

Nel secondo RRA si memorizzano 9600 letture eseguite ogni 15*4 secondi cioè 1 ogni minuto. Per u totale memorizzato di 160 ore.

Con il terzo RRA si archiviano 6000 letture memorizzando un valore ogni 24*15 secondi cioè ogni 6 minuti memorizzando in totale uno storico di 25 giorni.

Questo significa che dai tre archivi potremo analizzare cosa è successo negli scorsi 25 giorni con risoluzioni di 6 minuti, cosà è successo nelle ultime 160 ore con la risoluzione di 1 minuto e cosa è successo nelle ultime 40 ore con il dettaglio ogni 15 secondi.

Il totale delle letture dà la dimensione dell'archivio e in base all' heartbeat si ha la risoluzione e di conseguenza il periodo massimo monitorato.

Per ulteriori dettagli trovate ampie spiegazioni dei parametri nella pagina di manuale che si ottiene con

[root@jupiter root]# man **rrdcreate**

Finora abbiamo solo creato il database che va popolato con i dati da analizzare:

Creiamo ora un piccolo script che, in maniera abbastanza brutale, va a leggere il carico medio del sistema negli ultimi 1, 5, 15 minuti, il numero dei processi in escuzione/il numero dei processi totali, l'ultimo ID di processo assegnato dal sistema. Lo script estrae poi il carico dell'ultimo minuto memorizzandolo nel database.

Utilizzando un editor (io ho scelto vi)
[root@jupiter root]# vi av.sh

copiate lo script riportato di seguito. Penso sia sufficientemente commentato da evitare ulteriori spiegazioni.

```
#!/bin/bash
while [ 1 ] ; do
        echo "updating load.."
        echo ""

        #estraiamo il carico dell'ultimo minuto
        CURLOAD=`cat /proc/loadavg | cut -f 1 -d \ `

        #memorizziamo il valore ottenuto
        rrdtool update loadav.rrd N:$CURLOAD

        #diamo qualche informazione a video
        CURTIMEIS=`date`
        echo "updated at "$CURTIMEIS" with "$CURLOAD
        echo ""

        #attendiamo 10 secondi prima di ripetere il tutto
        sleep 10s
done
```

Ora trasformate il file in eseguibile ed avviatelo
[root@jupiter root]#   chmod +x av.sh
[root@jupiter root]#   ./av.sh
Riporto uno stralcio dell'output tagliato per evitarvi la monotonia di migliaia di righe sempre uguali
 [..]

updated at mer mag 10 16:28:22 EDT 2006 with 0.00

updating load..

updated at mer mag 10 16:28:33 EDT 2006 with 0.00

updating load..

updated at mer mag 10 16:28:43 EDT 2006 with 0.00

Dopo un po di tempo ho interrotto manualmente (ctrl+C) il programmino. Da buon curioso ho verificato che il file ha effettivamente dimensione fissa e riporta l'ora dell'ultimo aggiornamento.

[root@jupiter root]# ls -l

-rw-r--r-- 1 root root 202524 10 mag 16:29 loadav.rrd

Sempre da buon curioso ho voluto verificare come la dimensione del file fosse dipendente dai parametri di creazione. Chiaramente essendo tanto pigro quanto curioso non ho letto i sorgenti ma mi sono arrangiato con un paio di test per un po di reverse engineering. Di seguito vedete le istruzioni di creazione di 3 RRA con rispettivamente 1000, 2000 e 1 elemento e di seguito la dimensione del file ottenuto:

[root@giacomini mytests]# rrdtool create loadav2.rrd --step 10 DS:load:GAUGE:30:0:100
RRA:AVERAGE:0.5:1:1000

-rw-r--r-- 1 root root   8540 12 mag 12:11 loadav2.rrd

[root@giacomini mytests]# rrdtool create loadav2.rrd --step 10 DS:load:GAUGE:30:0:100

RRA:AVERAGE:0.5:1:2000

-rw-r--r-- 1 root root   16540 12 mag 12:11 loadav2.rrd

[root@giacomini mytests]# rrdtool create loadav2.rrd --step 10 DS:load:GAUGE:30:0:100 RRA:AVERAGE:0.5:1:1

-rw-r--r-- 1 root root   548 12 mag 12:13 loadav2.rrd

Dai numeri si evince che esiste un overhead fisso di 540 bytes dovuto probabilmente alle intestazioni ed ai puntatori, più 8 bytes per ogni dato:

8x1= 8 + 540=548

8x1000 + 540=8540

8x2000 + 540=16540

Ripetendo l'esperimento per due serie si ha un hoverhead è di 732 bytes (meno del doppio) che indica alcuni elementi in comune alle serie. Quindi effettivamente la dimensione del database è predeterminata alla creazione ed è pari al numero totale di elementi moltiplicati per 8 bytes. Al totale va aggiunto un overhead di 540 bytes per la prima serie (inferiore per le successive) il cui "peso" sulle dimensioni è ovviamente tanto minore quanto maggiore è il numero di elementi memorizzati.

Ora abbiamo ottenuto un database con dei dati. Sfrutto quindi il comando fetch per visualizzarne il contenuto che è mostrato di seguito troncato per leggibilità:

[root@jupiter root]# rrdtool fetch loadav.rrd AVERAGE

1147291750: nan

1147291760: nan

1147291770: nan

1147291780: nan

[..]

1147292000: nan

1147292010: nan

1147292020: nan

1147292030: nan

1147292040: 6.3000000000e-01

1147292050: 5.4000000000e-01

1147292060: 4.5800000000e-01

1147292070: 3.8700000000e-01

[..]

1147292910: 0.0000000000e+00

1147292920: 0.0000000000e+00

1147292930: 0.0000000000e+00

1147292940: 0.0000000000e+00

1147292950: 0.0000000000e+00

1147292960: nan

1147292970: nan

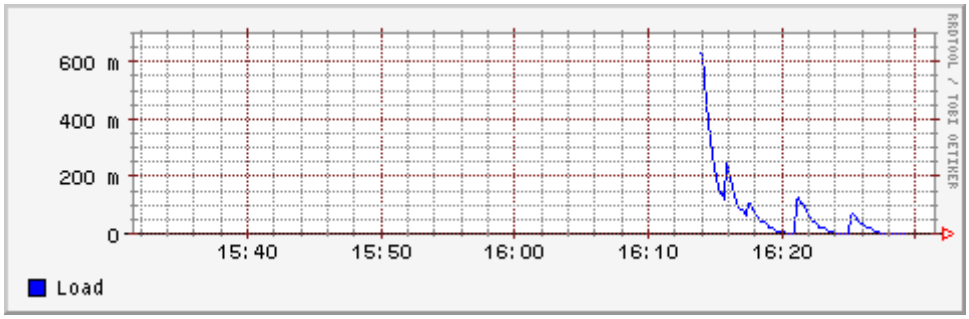
Visto così l'estratto delle righe memorizzate non ci trasmette un gran numero di informazioni, e il dump in XML non è molto più esaustivo anche se perlomeno contiene alcuni elementi relativi al DS che aiutano nella comprensione dei dati:

```
[root@jupiter root]#  rrdtool dump loadav.rrd
<!-- Round Robin Database Dump -->
<rrd>
        <version> 0001 </version>
        <step> 10 </step> <!-- Seconds -->
        <lastupdate> 1147292953 </lastupdate> <!-- 2006-05-10 16:29:13 EDT -->

        <ds>
                <name> load </name>
                <type> GAUGE </type>
                <minimal_heartbeat> 30 </minimal_heartbeat>
                <min> 0.0000000000e+00 </min>
                <max> 1.0000000000e+02 </max>

                <!-- PDP Status -->
                <last_ds> UNKN </last_ds>
                <value> 0.0000000000e+00 </value>
                <unknown_sec> 0 </unknown_sec>
        </ds>

<!-- Round Robin Archives -->
        <rra>
                <cf> AVERAGE </cf>
                <pdp_per_row> 1 </pdp_per_row> <!-- 10 seconds -->
                <xff> 5.0000000000e-01 </xff>

                <cdp_prep>
                        <ds><value> NaN </value>  <unknown_datapoints> 0 </unknown_datapoints></ds>
                </cdp_prep>
                <database>
                        <!-- 2006-05-09 13:49:20 EDT / 1147196960 --> <row><v> NaN </v></row>
                        <!-- 2006-05-09 13:49:30 EDT / 1147196970 --> <row><v> NaN </v></row>
                        <!-- 2006-05-09 13:49:40 EDT / 1147196980 --> <row><v> NaN </v></row>
                        <!-- 2006-05-09 13:49:50 EDT / 1147196990 --> <row><v> NaN </v></row>
                        <!-- 2006-05-09 13:50:00 EDT / 1147197000 --> <row><v> NaN </v></row>
                        <!-- 2006-05-09 13:50:10 EDT / 1147197010 --> <row><v> NaN </v></row>
                        <!-- 2006-05-09 13:50:20 EDT / 1147197020 --> <row><v> NaN </v></row>
                        <!-- 2006-05-09 13:50:30 EDT / 1147197030 --> <row><v> NaN </v></row>
```

```
                <!-- 2006-05-09 13:50:40 EDT / 1147197040 --> <row><v> NaN </v></row>
               [...]
                <!-- 2006-05-10 16:24:00 EDT / 1147292640 --> <row><v> 3.6250000000e-02 </v></row>
                <!-- 2006-05-10 16:28:00 EDT / 1147292880 --> <row><v> 1.9250000000e-02 </v></row>
            </database>
        </rra>
</rrd>
```

Quindi per avere un qualcosa di realmente utile ho creato il grafico:
[root@jupiter root]# rrdtool graph /var/www/html/loadav.png DEF:load=loadav.rrd:load:AVERAGE LINE1:load#0000ff:Load --start -1h
480x155

Avendo avuto l'accortezza di indirizzare il risultato sulla directory del web server locale del mio PC posso visualizzarlo con il browser (in alternativa potete indirizzare il grafico alla directory corrente e visualizzare il tutto con un visualizzatore grafico qualsiasi):

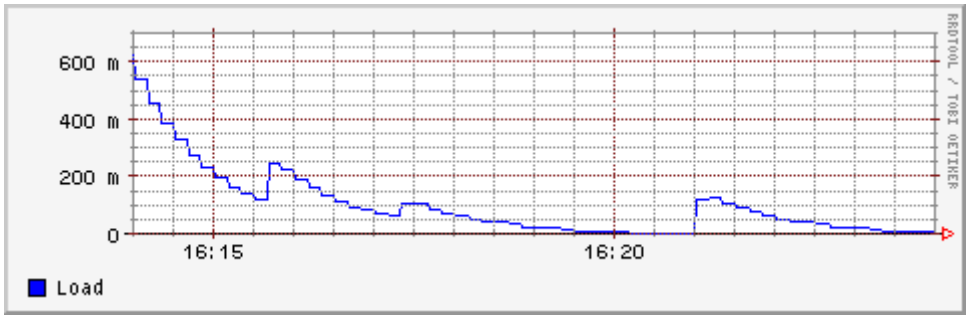


Volendo un intervallo temporale più ristretto (ovvero un maggior dettaglio):

[root@jupiter root]# rrdtool graph /var/www/html/loadav.png DEF:load=loadav.rrd:load:AVERAGE LINE1:load#0000ff:Load --start -40m --end -30m

480x155

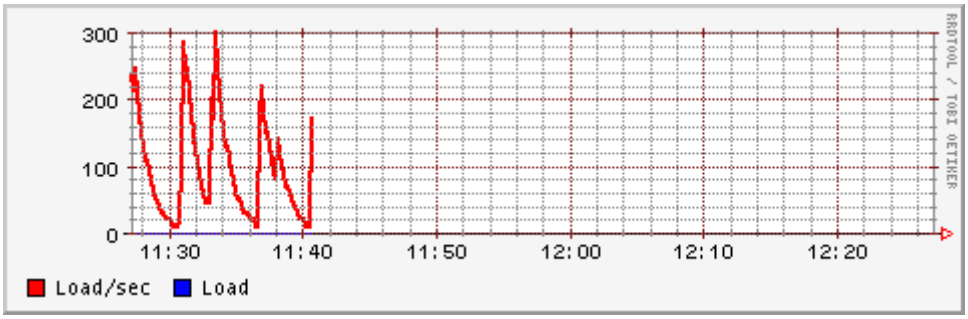
si ottiene



Vediamo ora un grafico su dati diversi con un po' di matematica all'interno:

[root@giacomini mytests]# rrdtool graph /var/www/html/loadav.png DEF:load=loadav.rrd:load:AVERAGE CDEF:loadsec=load,1000,* LINE2:loadsec#ff0000:Load/sec LINE1:load#0000ff:Load --start -1h

480x155



Ovviamente essendo il grafico del carico al secondo (rosso) su una scala 1000 volte maggiore a quella del carico in millisecondi (blu), quest'ultimo non risulta leggibile

Una nota: Quando si lavora con variabili di tipo contatore bisogna ricordarsi del fenomeno di riazzeramento (wrap), ovvero del fatto che il contatore arrivato al limite computabile (dipendente dal numero di bits che si intende usare nei conteggi) si azzera e ricomincia il conteggio. Rrdtool compensa automaticamente questo tipo di problemi quando la differenza tra due variabili è negativa ovvero quando si è presentato il fenomeno del passaggio per lo zero. Bisogna comunque fare attenzione di non lavorare con un basso numero di bits su intervalli di tempo troppo lunghi in quanto la nuova lettura dopo il passaggio per lo zero potrebbe assumere un valore più alto della lettura precedente. In tale caso la differenza sarebbe maggiore di zero e il riazzeramento non verrebbe intercettato dando luogo ad errori di misura. I valori di min e max possono essere utili per intercettare e scartare tali valori anomali.

Chiudo qui queste brevi note su RRDTool invitando coloro che trovano antipatica la riga di comando, a provare l'ottima interfaccia grafica per questo tool fornita da CACTI: un progetto open source nato appunto per semplificare l'approccio a RRDTool.

OCS Inventory

Nei capitoli precedenti ho illustrato vari sistemi atti al monitoraggio della rete al fine della individuazione dei guasti o della rilevazione di misure sulla stessa. La gestione dei sistemi informativi di una azienda, nella sua accezione più ampia, deve però includere anche altre modalità di monitoraggio delle risorse sia fini di una localizzazione rapida delle stesse, sia per necessità di tipo amministrativo, sia per ottimizzarne lo sfruttamento.

Tali attività vengono solitamente indicate con il termine inglese di *inventory* che può essere letteralmente tradotto in italiano come *inventario*. Lo scopo è, appunto, di inventariare le risorse: PC, servers, stampanti, apparecchiature varie, con le loro caratteristiche e, l'eventuale, software in esse contenuto. Se possibile, è utile che tali elenchi vengano mantenuti allineati con l'elenco degli utilizzatori di ciascuna risorsa in modo quanto più possibile automatico.

A tale scopo si utilizzano, per le apparecchiature collegate in rete, dei software che automatizzano il più possibile il processo di raccolta e aggiornamento di tali informazioni e che conservano tali dati in un database per future consultazioni e analisi, ed è di uno di tali software che andremo a parlare.

Il contesto operativo che verrà descritto è quello comune a molte aziende italiane nelle quali si opera all'interno di un Dominio con un Primary Domain Controller con Sistema Operativo (S.O.) Microsoft, La maggior parte dei client sono dotati di un S.O. dello stesso produttore (i client Open Sorce sono quasi sempre una minoranza, quando non sono addirittura assenti) e vi sono una serie di server aggiuntivi, quasi sempre con S.O. Open Source (molto spesso GNU/Linux), che sono in genere server applicativi. Chiudono l'elenco una serie di dispositivi di rete quali ad esempio stampanti, switch, print servers, firewalls, etc.

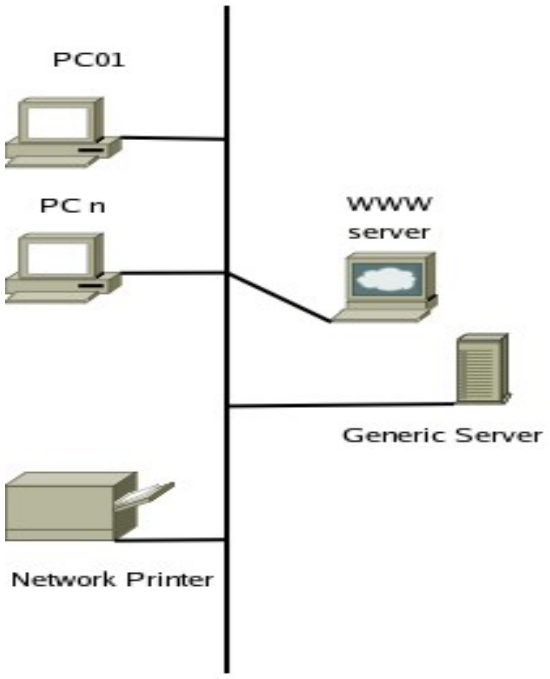


Fig.1 Rete generica

Il primo programma provato, H-Inventory[http://www.h-inventory.com/], pur avendo maggiori funzionalità di gestione e migliori reports (rispetto allo strumento scelto) non permette di personalizzare “al volo“ la home page, e una delle nostre necessità era che gli indirizzi IP dei computer inventariati fossero immediatamente disponibili. Inoltre H-Inventory prevede la rilevazione/trasmissione dei dati tramite condivisione di una cartella (smb) sul server o via FTP. Entrambi i servizi non erano e non sono disponibili nella macchina da noi prescelta per l'installazione del server di amministrazione e questo ha contribuito a far scegliere OCSInventory NG come alternativa.

OCSInventory NG [http://www.ocsinventory-ng.org/] ovvero Open Computer and Software Inventory Next Generation è un programma distribuito sotto licenza GPL v2 che permette di inventariare i computer della rete raccogliendo informazioni sull'hardware, il sistema operativo ed il software installato, di, distribuire software e di esplorare la rete alla ricerca di dispositivi.

Prevede un architettura client-server con un server centrale di raccolta dati (di fatto un server web) e un programma detto Agent che gira come servizio sui client.

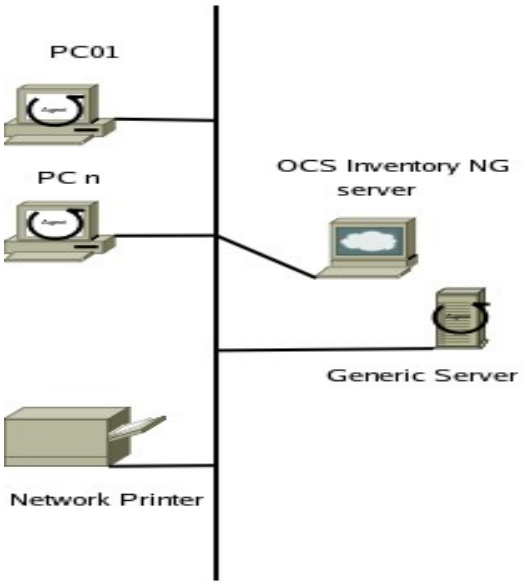


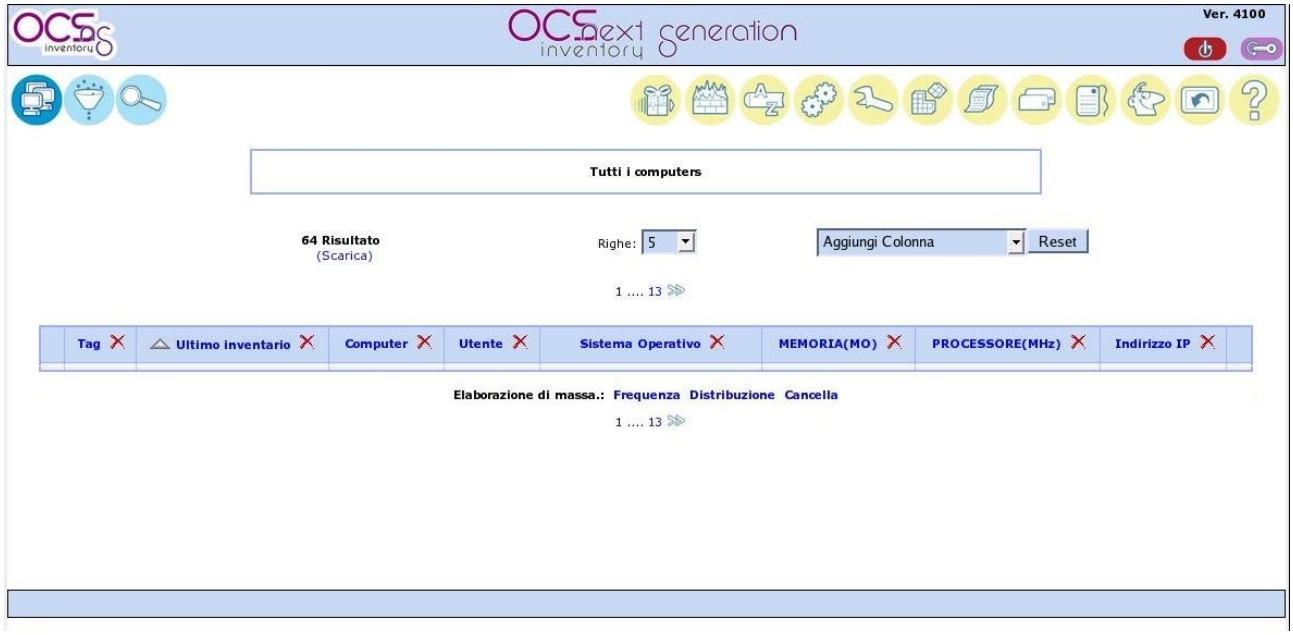
Fig. 2 Rete con OCS Inventory NG

Per il server i prerequisiti richiesti sono: un web-server Apache (v.1.3.33 o maggiore) con supporto a PHP (v.4.3.2 o maggiore) e MySQL 4.1.0 (o successiva) oltre ad un certo numero di moduli PERL come da manuale di installazione.

L'installazione (dopo avere soddisfatto le dipendenze) consiste semplicemente nel decomprimere il file .tar.gz in una directory del server web (nel percorso del server HTTP ovviamente), nel nostro esempio il file è stato decompresso nella web-root e la directory si chiama ocsreports. Fatto questo ci si porta all'interno di essa e si richiama il programma setup.sh. Il tutto è descritto in maniera chiara nel manuale [http://prdownloads.sourceforge.net/ocsinventory/OCS_Inventory_NG-Installation_and_Administration_Guide_1.9_EN.pdf.zip?download] per cui ritengo opportuno evitare di riscrivere questi passaggi.

Dopo l'installazione in un browser digitate l'indirizzo della directory nella quale avete eseguito l'installazione (nell' esempio: http://miositoweb/ocsreport/) e vi verrà richiesto di autentificarvi con login e password prescelti.

Una volta loggati vi si aprira la finestra principale nella quale ovviamente non è ancora riportato alcun dato:



Per trasmettere i dati di inventario dalla workstation al server OCSInventory usa un programma agent. Esistono agent sia per windows che per Linux.

Visto che uno degli obbiettivi, di questo tipo di gestione, è quello di ridurre al minimo le operazioni manuali, sono state previste varie modalità per la distribuzione degli agent evitando di dover effettuare l'installazione su ogni singolo computer.

Per l'installazione dell'agent in un dominio con un Primary Domain Controller Microsoft, la modalità più semplice è depositare il programma di installazione in una cartella condivisa e poi lanciare dallo script di logon l'apposito programma OcsLogon.exe che eseguirà l'installazione. Tale programma va preventivamente rinominato con il nome canonico del server sul quale risiede il programma di amministrazione o con il suo indirizzo IP. Ad esempio, ipotizzando di avere il programma di amministrazione installato, sul server, all'indirizzo 172.20.1.4 il programma OcsLogon.exe verrà rinominato in 172.20.1.4.exe ed il comando da inserire nello script di login sarà qualcosa tipo

"\\server\shared_dir\172.20.1.4.exe /DEBUG /NP /INSTALL"

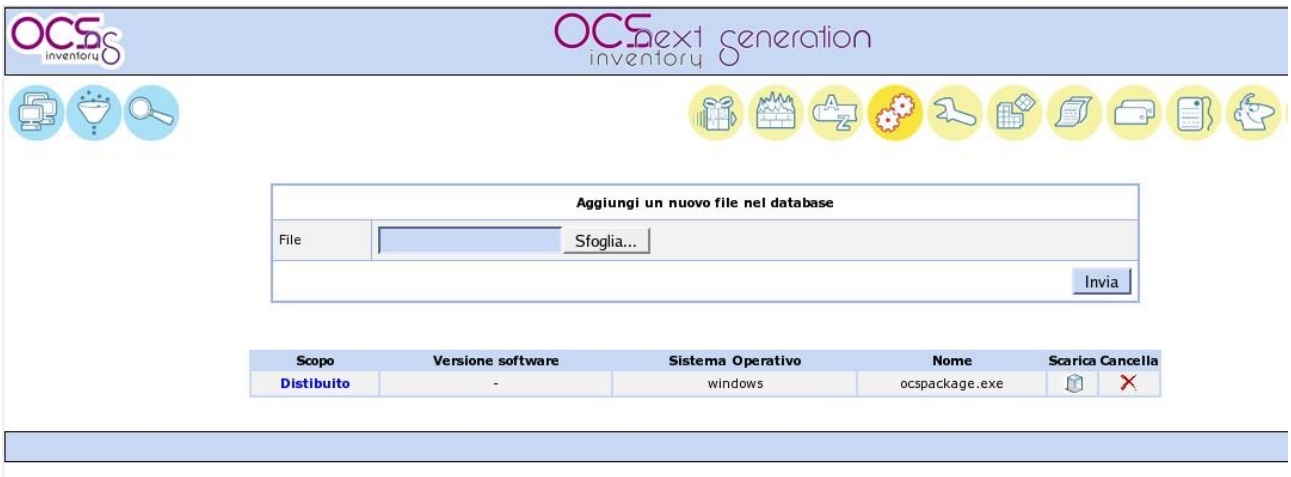
I parametri in coda hanno il seguente significato:

/DEBUG= traccia su un file le operazioni eseguite (utile in caso di problemi)

/NP= impone di non utilizzare il proxy della connessione HTTP (si suppone che il server di amministrazione sia in rete locale)

/INSTALL= esegue l'installazione dell'agente come servizio se ancora non è installato, altrimenti avvia solamente il servizio.

Da notare che l'eseguibile per l'installazione dell'agent, OcsAgentSetup.exe, scaricato dal sito non è immediatamente distribuibile "così com'è" ma ne va creata una versione "pacchetizzata" che va preventivamente predisposta, parametrizzata e caricata sul server di amministrazione,



infatti il programma OcsLogon.exe, appena visto, non riesce a passare alcun parametro al programma di installazione dell'agent se non la locazione del pacchetto è (indirettamente tramite l'aver rinominato l'eseguibile) l'indirizzo del server di amministrazione.

Per la preparazione del pacchetto da distribuire si utilizza un'altro programma distribuito con la suite: ocspackage.exe. Questo programma per windows dopo l'avvio chiede il nome dell'eseguibile che va usato per il setup dell'agent (OcsAgentSetup.exe), il percorso per un eventuale certificato, il nome e la password dell'amministratore di dominio (per conto del quale verrà eseguita l'operazione di installazione) e i parametri con i quali viene lanciato il setup.

Nel nostro caso i parametri di setup scelti sono /S /NP /DEBUG /SERVER:172.20.1.4

Il parametro S server per il "silent mode" in modo da non "disturbare" l'utente durante l'installazione gli altri parametri hanno lo stesso significato di quelli omonimi dell'OcsLogon.

L'intera sequenza, spiegata a parole, risulta abbastanza caotica per cui penso sia opportuno riassumere il tutto graficamente:

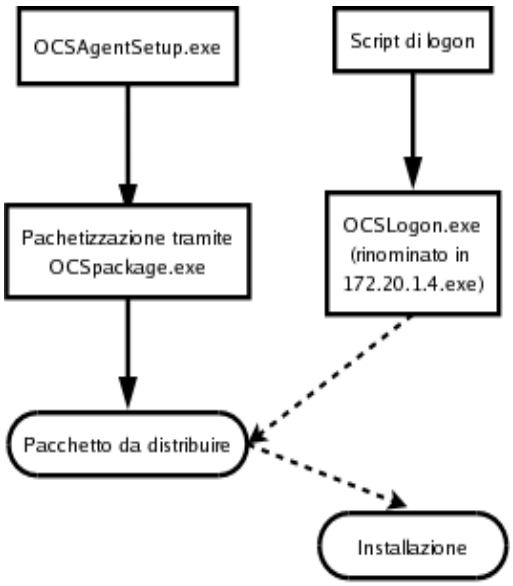


Fig. 3 Creazione del pacchetto dell'agent
e successiva installazione automatica

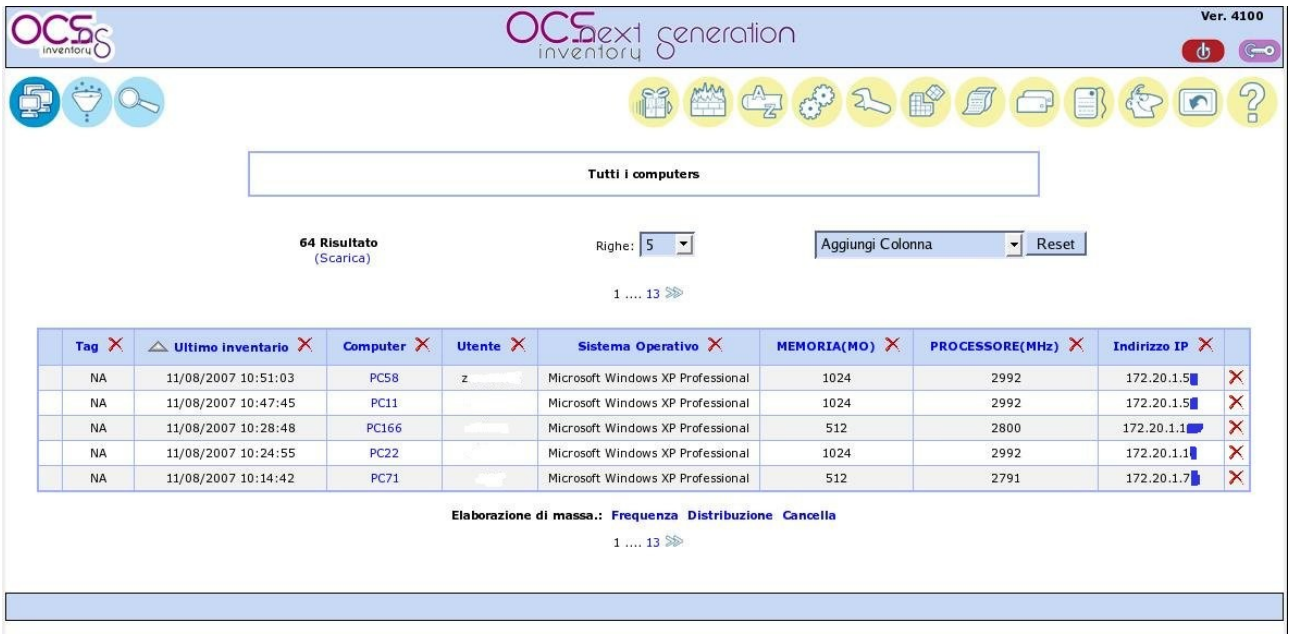
La frequenza con la quale l'agente comunica i dati al server è un parametro settato sul server, nella pagina di amministrazione del programma, che si chiama PROLOG_FREQ=xx.

Nome	Valore	Nome	Valore
AUTO_DUP	Serial, macaddress — Definisce ii criterio secondo il quale due computers devono essere uniti automaticamente.		ON / OFF
DOWNLOAD	ON / OFF	DOWNLOAD_CYCLE_LATENCY	60
DOWNLOAD_FRAG_LATENCY	10	DOWNLOAD_PERIOD_LATENCY	0
DOWNLOAD_PERIOD_LENGTH	10	DOWNLOAD_TIMEOUT	30
FREQUENCY	ALWAYS / NEVER / CUSTOM	INVENTORY_DIFF	ON / OFF
INVENTORY_TRANSACTION	ON / OFF	IPDISCOVER	ON / OFF — 4
IPDISCOVER_LATENCY	100	IPDISCOVER_MAX_ALIVE	7
LOCAL_PORT	80	LOCAL_SERVER	http://172.20.1.4
LOGLEVEL	ON / OFF	PROLOG_FREQ	6
REGISTRY	ON	TRACE_DELETED	ON

Tale parametro viene utilizzato come seme per ottenere un valore random con xx come massimo (nell'immagine di esempio xx=6 ore). Lo scopo della randomizzazione è evitare che tutti i client vadano ad inviare contemporaneamente i dati al server congestionando il sistema. Al primo collegamento l'agente scarica tale valore e lo sincronizza in locale. Il valore xx rappresenta il tempo massimo in ore entro il quale l'agente invierà i dati.

Per verificare che l'agente sia operativo è sufficiente verificare il file service.ini nella directory del programma. Tale programma riporta nel parametro TTO_WAIT=yyyy il valore in secondi mancante all'invio dati al server. E' semplice verificare a pochi secondi di distanza se tale parametro è variato del valore atteso (il valore iniziale meno i secondi trascorsi, ovviamente). Quando il conteggio scende a zero l'agent, tramite il protocollo HTTP, invia i dati al server che li memorizza nel database.

A questo punto è possibile accedere al server e visualizzare i dati raccolti.



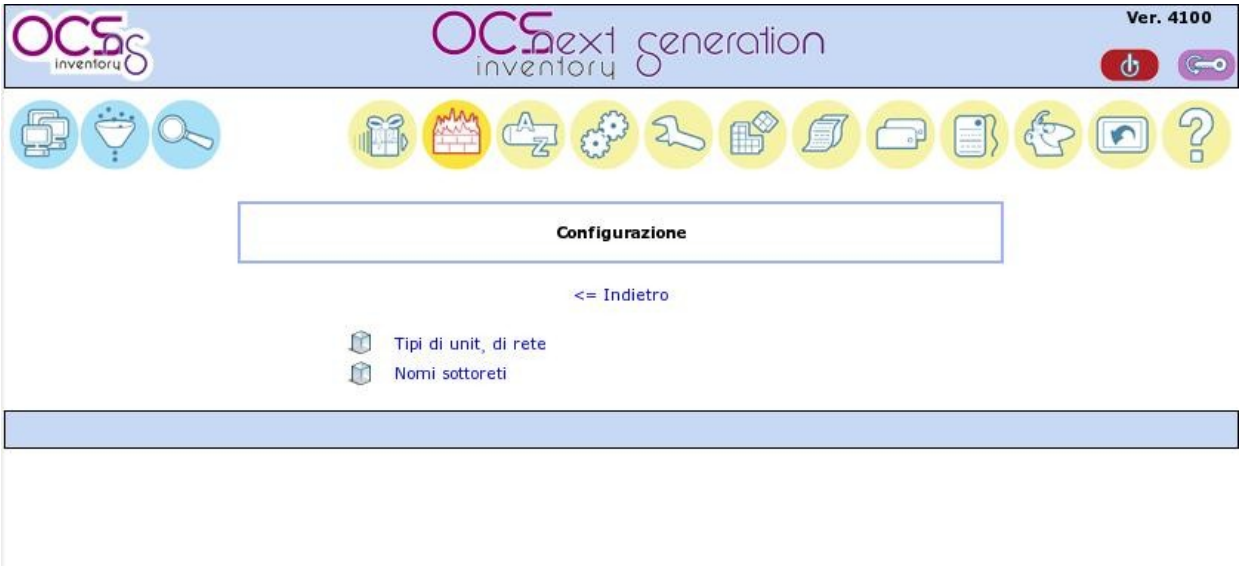
Le operazioni di discovery

In una rete, ovviamente, non ci sono solo i PC client ma anche una serie di dispositivi sui quali l'agent non può essere installato. Per catalogare tali dispositivi entrano in gioco le funzionalità di “discovery” di OCSInventory. E' sufficiente settare su ON il parametro IPDISCOVER nella pagina di amministrazione sul server. Accanto a tale parametro è riportato un numero che indica (come vedremo meglio di seguito) il numero di client che verranno coinvolti nell'operazione di discovery.

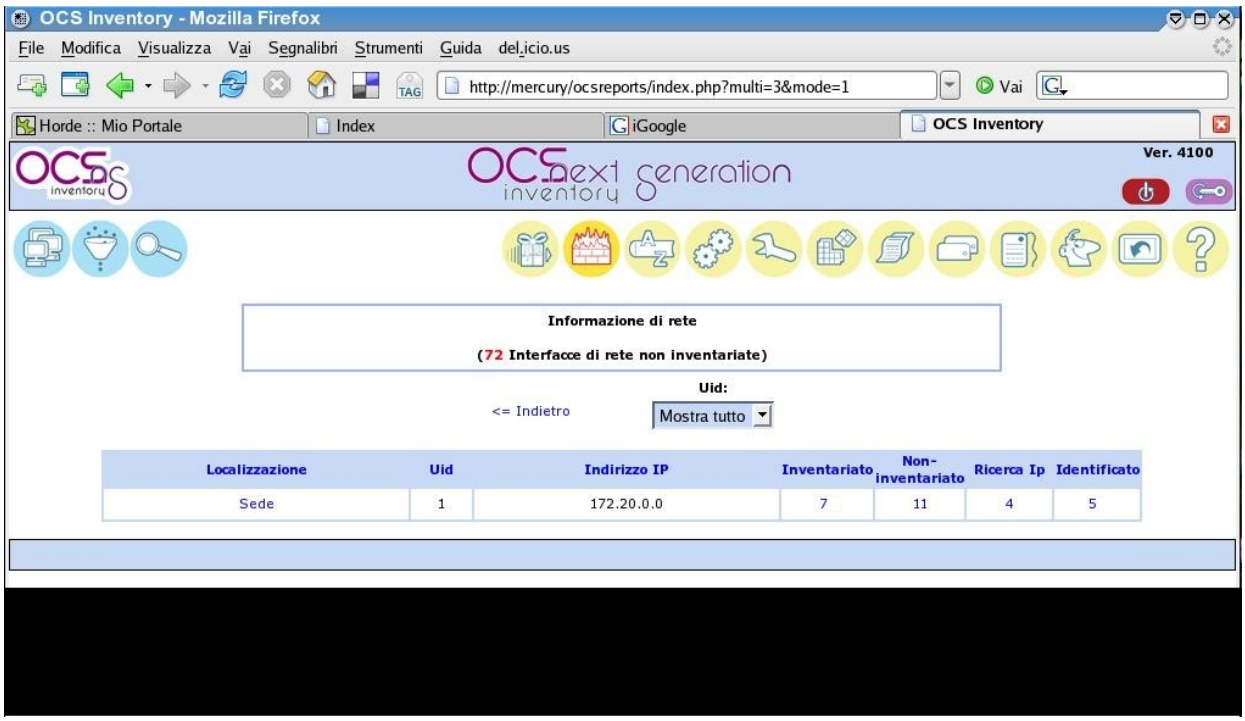
In pratica il server centrale, basandosi sulla assiduità con la quale inviano informazioni, incarica il numero indicato di client di scandagliare le reti definite. Qualora un ip risponda all'interrogazione viene memorizzato come ip da identificare.

Quindi i passi da eseguire per attivare la rilevazione sono:

1) Definire le reti da rilevare



2) Abilitare il parametro IPDISCOVER nella finestra di amministrazione del server
3) Attendere il rilevamento
4) Identificare gli IP rilevati che vengono classificati come non inventariati nella finestra delle informazioni di rete (nell'immagine seguente sono 11)



Vista la sua semplicità non vorrei dilungarmi oltre nella descrizione di questo programma supportato, oltretutto, da un buon manuale in inglese. Lo scopo dell'articolo era di portare alla vostra attenzione uno strumento che riguarda un'attività molto spesso trascurata da noi informatici ovvero l'inventario delle proprie risorse.

Come ho premesso fin dall'inizio ritengo, però, che una corretta gestione delle risorse disponibili possa rappresentare quel “di più” che fa la differenza fra una gestione professionale ed una “improvvisata” di un dipartimento informatico.

Gestione sistemi - conclusioni

A novembre 2005 iniziavo a scrivere le prime note sulla gestione dei sistemi, e queste note derivavano da un lavoro ancora precedente. Sono passati 3 anni e, come si usa dire, tanta acqua sotto i ponti. Visto, poi, che per l'informatica 3 anni corrispondono ad un era direi che è arrivato il momento di tirare le somme e fare il punto della situazione.

Nagios

Il tutto è iniziato in settembre od ottobre del 2005 stavo installando Nagios, con l'obiettivo di monitorare un paio di server critici, e mi trovavo in difficoltà con la documentazione in formato HTML che mi costringeva a saltare continuamente da un capitolo all'altro, e, quindi, da una pagina web all'altra, per seguire i concetti espressi. Per cui avevo deciso di copiare tutto in un unico documento Open Office e formattarlo per benino in modo da avere a disposizione un manuale stampato più agevole da consultare. Ne è uscito un file, in formato PDF, che avevo offerto come contributo ad Ethan Gastald (il creatore di Nagios) solo per scoprire che aveva già incaricato una persona di fare questo lavoro...

Per non sprecare il lavoro fatto, che comprendeva anche delle parziali traduzioni, ho integrato queste ultime con le note di installazione (che nel frattempo avevo completato) e con alcuni estratti del manuale e ne ho ricavato un bell'articolo per il Pluto Journal. Viste le dimensioni dello scritto, l'articolo, è stato spezzato in due e la serie è poi continuata di pari passo con i miei lavori successivi.

Nel frattempo Nagios è cresciuto e dalla versione 2.0 siamo ora alla 3.0.3 che rende parzialmente inutili le mie note essendo dotata di una sezione di quickstart per l'installazione rapida nelle distribuzioni più comuni.

Le istruzioni e i concetti di base rimangono comunque validi e li ritengo ancora una buona base per chi volesse affrontare questo programma.

Nel frattempo ho visto un progetto analogo crescere e diventare valido: Big Sister Network Monitor [http://www.bigsister.ch/projects.html]. Non ho avuto modo di provarlo direttamente ma ne ho sentito parlare bene per cui vi invito a valutarlo come alternativa.

SNMP

Il terzo articolo della serie riguardava l'SNMP. Visto che non vi si menzionava alcun prodotto in particolare direi che le informazioni contenute rimangono valide. Purtroppo sento parlare sempre meno di questo protocollo che, comunque, viene ancora sfruttato da molti software per la gestione dei sistemi ed è presente in moltissimi dispositivi in vendita.

RRDTool

Oggetto del quarto articolo della serie, questo strumento, è arrivato alla versione 1.3. Penso che nessuno lo stia utilizzando da linea di comando nel modo da me descritto nell'articolo. Infatti al termine dello stesso concludevo segnalando il programma Cacti. Cacti è un ottima interfaccia web-based, che trasforma RRDTool in un comodo e ottimo strumento di analisi, alla quale avrei voluto dedicare un articolo di questa serie o, perlomeno, uno spazio maggiore all'interno dell'articolo di RRDTool. La mancanza di tempo ha fatto si che il mio uso, di tale interfaccia, si risolvesse ad un test di alcuni giorni sul quale non ho avuto modo di scrivere molto. Ho ritenuto superfluo riportare la procedura di installazione e configurazione da me utilizzate all'epoca in quanto ho semplicemente seguito il manuale. Anche i risultati della mia prova pratica (un test su una tratta della LAN), per quanto utili non lasciavano spazio a molti commenti e considerazioni.

Altri strumenti similari che utilizzano RRDTool e che non ho ancora avuto modo di testare sono:

Smokeping [http://oss.oetiker.ch/smokeping/] per l'analisi delle latenze di rete

MRTG [http://oss.oetiker.ch/mrtg/] per l'analisi del traffico dei router

Entrambi, fra l'altro, sono stati sviluppati proprio dallo stesso programmatore che ha creato RRDTool.

OCSInventory NG

L'ultimo articolo in ordine di tempo riguardava un programma di inventario ed Ã¨ comparso nello scorso numero del Pluto Journal. Essendo stato scritto in tempi relativamente recenti eviterei di tornarci sopra.

Tkined

La serie si doveva chiudere con un articolo su Tkined (rpm a questo link: http://rpm.pbone.net/index.php3/stat/4/idpl/4387636/com/scotty-tkined-3.0.0-0.20030629.1mdk.i586.rpm.html). Questo programma scritto in Tcl/Tk è praticamente un coltellino svizzero per l'amministratore di rete. Permette di disegnare una rete sia manualmente sia tramite una modalità di autorilevamento. La "mappa" ottenuta può essere resa "attiva"

indicando quali computer/server/dispositivi vanno monitorati e quali test devo essere eseguiti.

Il programma ha molte potenzialità ma anche molti difetti:

- l'interfaccia è, a dir poco, spartana e minimalista.
- l'installazione e un incubo di dipendenze e rintracciare i pacchetti necessari è un impresa
- cosa peggiore: il progetto sembra abbandonato.

Tutto ciò mi ha spinto a non dedicare troppo tempo a questo programma anche se, personalmente, continuo a sfruttare le sue funzionalità di disegno. Spero fortemente che qualcuno possa riprendere lo sviluppo di questo tool o di uno similare in quanto si sente decisamente la mancanza di alcune funzionalità da esso implementate.

Conclusioni

Chiudo quindi questa serie di articoli con un piccolo rimpianto. Secondo le mie intenzioni iniziali doveva essere molto più articolata e lunga ma avevo bisogno di collaborazione soprattutto per verificare e testare alcuni programmi che non ho tempo e/o possibilità di provare personalmente.

Ho lanciato varie volte appelli in questo senso ma non sono stati raccolti quindi non mi resta altro da fare che chiudere questa bella esperienza ringraziando tutti coloro che mi hanno scritto complimentandosi per il lavoro svolto finora.

Riferimenti

Nagios:

Manuale di Nagios [http://nagios.sourceforge.net/docs/2_0/]

Installing and configuring Nagios by Kate Harris at TOTKat site [http://www.totkat.org/pages/nagios.shtml]

...i sorgenti di Nagios (of course)

SNMP:

Essential SNMP di Kevin Schmidt, Douglas Mauro – O'Reilly - ISBN: 0596000200

RFC 1157 su SNMP [http://www.faqs.org/rfcs/rfc1157.html]

RFC 1351: SNMP Administrative model [http://www.faqs.org/rfcs/rfc1351.html]

RFC 1441 Introduction to version 2 of INMF [http://www.faqs.org/rfcs/rfc1441.html]

RFC 1442 Structure of MIF for SNMP v2 [http://www.faqs.org/rfcs/rfc1442.html]

NET SNMP FAQ [http://www.net-snmp.org/docs/FAQ.html]

The Simple Times magazine:[http://www.simple-times.org/]

RRDTool:

RRDtool Demystified di Bharat Shetty – O3 Magazine n.4 [http://www.o3magazine.co.uk/o3/issue4/o3-i4-72.pdf]

RRDTool Tutorial [http://oss.oetiker.ch/rrdtool/tut/rrdtutorial.en.html]

RRDTool Home page [http://oss.oetiker.ch/rrdtool/]

Getting Started with RRDtool [http://cuddletech.com/articles/rrd/index.html]

Una discussione su xff, heartbeat e step [http://lists.ee.ethz.ch/rrd-users/msg03115.html]

OCS Inventory NG:

- Il manuale di installazione e amministrazione [http://prdownloads.sourceforge.net/ocsinventory/OCS_Inventory_NG-Installation_and_Administration_Guide_1.9_EN.pdf.zip?download].

Indice generale