
# Università degli studi di Pisa
# Esame di Sistemi per l'elaborazione dell'informazione: Gestione di Rete.
# Un'architettura distribuita per il management di reti GSM.

De Col Daniela

5 luglio 2001

# 1 Analisi del problema

## 1.1 Il problema

Si richiede la realizzazione di un'architettura di management per un sistema di gestione GSM [GSM] distribuito su tre aree (Nord, Centro e Sud). In particolare sono ritenute importanti le informazioni che riguardano il servizio erogato.
In questa particolare ottica del problema si possono individuare da subito le aree funzionali proprie dell'applicazione di management richiesta: Gestione delle configurazioni per quanto riguarda la localizzazione, gestione degli account per definire costi e consumi degli utenti, gestione della performance per quanto riguarda il mantenimento di dati statistici quali: l'utilizzo percentuale dei servizi e la distribuzione dei clienti nelle tre zone. È però opportuno pensare ad un possibile ampliamento delle funzionalità di management della nostra applicazione che potrà essere utilizzata per la gestione dell'apparato GSM in tutte le sue parti (sistemi fisici che erogano il servizio, applicazioni, reti, ecc.)
In questo caso verranno gestiti anche i malfunzionamenti tipici delle reti fisiche e quelli dei protocolli che possono rivelare problemi interni o tentativi di intrusione dall'esterno. Si terrà inoltre conto, per quanto riguarda la gestione della configurazione, delle informazioni sulla distribuzione dei vari servizi sul territorio (ricordo che noi ci limitiamo al servizio di chiamata), della determinazione dei tempi di utilizzo dei servizi e delle scadenze. La gestione della configurazione sarà affiancata dalla gestione della performance che permetterà, mediante l'analisi delle statistiche di modificare, la configurazione dei servizi per garantire una migliore erogazione e quindi aumento di utili per l'azienda.

La distribuzione del management, come già detto, è su tre aree, per questo motivo si può pensare ad una architettura gerarchica (vedi figura 1) che prevede, in base al paradigma *manager/agent* [Management], tre manager (uno per ogni zona) che si occupano della gestione locale e un manager centrale che si limita alle direttive e verifiche globali, cosicché solo i management di zona gestiscono effettivamente gli agent.
È facile prevedere una semplice estensione dell'architettura a più aree e quindi una sua estensione o l'utilizzo anche in altri ambiti di gestione di agenti mobili.

Gli agent interagiscono con le risorse reali per implementare i *managed objects* (MOs) [Management] e i manager per immagazzinare le informazioni utili al management, nella sezione 2 vedremo come.

## 1.2 I managed object

Per definire un'architettura di management è necessario prima di tutto individuare quali sono i managed object (MO) da gestire.

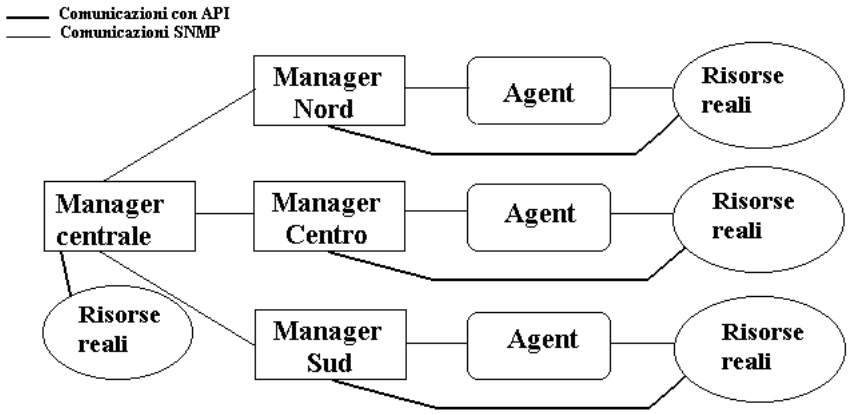


Figura 1: Architettura di base

Nel nostro caso è opportuno associare ad ogni utente GSM un MO che riporti le informazioni interessanti per il management. Questi MO saranno a lunga durata (ovvero il lasso di tempo tra creazione e cancellazione sarà considerevole), mentre si possono prevedere dei MO a breve durata che riportino gli attributi interessanti delle conversazioni. Vediamoli più in dettaglio definendone: gli attributi, le operazioni, il comportamento e le notifiche.

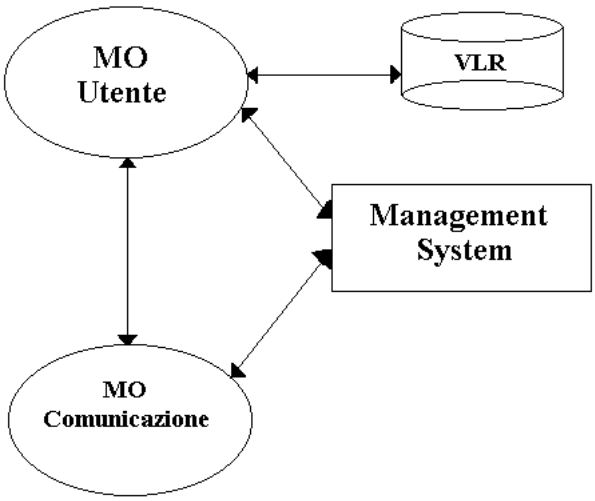


Figura 2: I managed objects

### 1.2.1 Gli attributi

Gli attributi degli utenti sono: indicativo SIM[1], numero di telefono, nome, cognome, BSC[2] attuale, data e ora di accesso alla BSC, servizi attivi. Per le comunicazioni: numero chiamante, numero chiamato, data e ora di inizio, data e ora di fine.

### 1.2.2 Le operazioni

Le operazioni quì riportate sono le tipiche operazioni su MOs ma vedremo in seguito che nella realizzazione degli agent si dovrà tener conto della grande quantità di MOs da gestire e quindi dell'impossibilità di definire operazioni così semplici (ovvero le operazioni che ora possono sembrare semplici verranno esplicitate con l'utilizzo del naming).

- *La creazione*
  questa operazione viene effettuata quando un GSM richide la registrazione in una delle tre aree e, per le comunicazioni, quando queste vengono effettuate da un GSM in una determinata area.
- *La cancellazione*
  simmetricamente alla creazione, la cancellazione verrà eseguita quando un GSM esce dall'area di management. Allo stesso modo quando viene chiusa una comunicazione il MO associato viene cancellato.
- *La set*
  è l'operazione che permette la modifica degli attributi del MO.
- *La get*
  permette la lettura degli attributi del MO.

### 1.2.3 Il comportamento

**MO utente** I MOs riferiti agli utenti devono interagire con il DB VLR (Data Base Visitor Location Register) [GSM] per verificare gli accessi degli utenti (creazione, cancellazione) e con il management system reagendo alle richieste e segnalando anomalie. A questo proposito è necessario prevedere un DB o una specie di log file per il collezionamento di informazioni storiche utili per le scelte di configurazione e business planning.
Dato che l'architettura è piramidale anche i DB saranno distribuiti sulle tre aree, ottenendo tre DB di zona per i submanager e un DB per il manager centrale. Ogni DB di zona conterrà tutte le informazioni storiche della zona, cioè tutte quelle necessarie per per definire la gestione locale. Nel DB centrale saranno eventualmente duplicati i dati più frequentemente acceduti (una

[1]Subscriber Identity Module
[2]Base Station Controller

cache che diminuisca il traffico di informazioni sulla rete) e le informazioni per il management globale.

**MO comunicazione** I MO riferiti alle comunicazioni sono strettamente legati ai precedenti in quanto sono creati da azioni dei GSM. Anch'essi sono legati al VRL dal quale ricavano informazioni sul GSM che le ha attivate ed iniltre si servono del DB locale per inserirvi i dati prima della cancellazione, si occupano infine di comunicare le informazioni importanti per la tariffazione al sistema di billing (non si può escludere che il sistema di billing prelevi tali informazioni direttamente dal DB locale).

### 1.2.4 Le notifiche

Una caratteristica importante dei MOs è quella di poter inviare delle notifiche (definibili con TRAP SNMP) su avvenimenti importanti o fuori dal normale.
Queste situazioni degne di nota vengono accuratamente definite dal Manager e sono poi utilizzate come mezzo di controllo del sistema.

Nel nostro caso possiamo prevedere delle notifiche per la creazione e cancellazione di MOs utente in quanto può essere utile sapere quando un nuovo utente si registra nell'area e quando si deregistra, inoltre in questo modo si limitano le operazioni di polling del manager locale.
Possono essere inoltre definite delle notifiche che informino quando il numero di utenti per area va sotto una certa soglia predefinita. TRAP di questo tipo sono importanti per avere informazioni sull'andamento del servizio erogato e sul buon funzionamento delle apparecchiature fisiche. Per esempio un numero di utenti connessi molto inferiore al solito potrebbe indicare che alcune apparechiature hanno smesso di funzionare corretamente o che un'altra compagnia stà applicando una tariffa migliore o offre addirittura un servizio migliore. Accorgersi in tempo di questi eventi è importante e le notifiche devono essere accuratamente progettate prima di essere utilizzate perché potrebbero portare a falsi allarmi [TRAP].

# 2 Progettazione

A questo punto per definire in dettaglio l'architettura è necessario fare una scelta sui linguaggi di implementazione dei MIB e sui protocolli di accesso ai MO remoti poiché queste scelte hanno un impatto importante sull'efficienza e sulla complessità del sistema di management nonché sulla sicurezza.

## 2.1 Scelte di progetto

Data la particolarità degli oggetti trattati, agenti mobili, è necessario pensare ad una implementazione che renda le operazioni di management veloci

(ci saranno molti cambiamenti di stato dei MO), inoltre la grande quantità di oggetti da gestire porta ad un appesantimento del sistema: i dati occupano molta memoria e quindi risorse di rete durante le richieste di accesso e modifica. Non vogliamo inoltre applicativi troppo diversi tra loro perché richiedono interfaccie di adattamento per lo scambio dei dati che rallentano a loro volta il sistema.
Quindi meglio un unico linguaggio interfacciabile con il protocollo SNMP [Management], a questo scopo sembra plausibile l'utilizzo di *Java* [Java] che nelle sue varie forme permette di definire manager, agent e interfaccie grafiche. Java è portabile e cioé indipendente dalla piattaforma che abbiamo a disposizione e che potrà essere necessario sostituire nel tempo.

Le diverse aree di applicazione del management in un ambiente così grande come un'azienda di servizi sono riferite alle necessità di analisi di diverse persone (o gruppi di persone) e non è opportuno che tutto sia visibile a tutti. Bisogna perciò tener conto anche della necessità di definire delle barriere di sicurezza al sistema.

Si devono prevedere delle "viste di accesso" al sistema come quelle previste dal protocollo SNMPv3 [Management] che da la possibilità di indicare tali viste già a livello di definizione del MIB. Le viste permettono di suddividere le informazioni in modo tale che solo chi è autorizzato a leggerle o ad eseguirvi delle operazioni ne sia effettivamente capace. In questo modo si evita che le informazioni vengano viste da chiunque ma soprattutto che esse vengano modificate da chi non ne ha le competenze, evitando possibili danni al sistema. Questi problemi sono interni all'azienda e riguardano le suddivisioni di compiti tra coloro che vi lavorano.

Oltre ai problemi di sicurezza interni vi sono poi quelli esterni, ancora più problematici e che richiedono una trattazione di estrema cautela, parliamo in questo caso di possibili attacchi da parte di hacker ma anche di problemi di spionaggio industriale.

Java ci mette a disposizione dei meccanismi di sicurezza come il Java security manager, che garantisce la corretta esecuzione delle applicazioni via web senza permettere la modifica dei dati. È comunque necessario definire delle regole di accesso ai dati e alle applicazioni (gestione delle password).

### 2.1.1 Gli agents

Per la realizzazione dei MOs ho pensato a degli script agents [Management] che possono facilmente essere modificati, garantendo così un management che si evolve assieme ai servizi gestiti (i servizi GSM sono in continua evoluzione).

È possibile scrivere gli agents in Java e utilizzare le api JMX [Java] per l'interfacciamento con SNMP così la definizione dei MOs nel MIB SNMP è molto semlice.
Dato che ogni agent deve gestire milioni di MOs non è plausibile immaginare

che tutti gli attributi vengano definiti nei mib del MO. Per questo motivo introduciamo la tecnica del *naming* [Management] che useremo come avviene già per le tabelle di routing.
Costruiamo una tabella formata da due colonne, nella prima sarà indicato il nome dell'istanza di MO che ci interessa. Tale nome è composto da due parti: l'Object identifier, cioè la sequenza di numeri che identifica la sua posizione nel MIB, e l'instance identifier che identifica l'istanza dell'oggeto, nel nostro caso sarà il numero di telefono del GSM. La seconda colonna della tabella conterrà una specie di puntatore ai dati riferiti a quell'istanza che si trovano nel DB locale.

| La tabella di naming | |
|---|---|
| Object ID . Instance ID | Puntatore |
| 1.3.6.4.12.2.3.4.7.2.1.2.3.2.4.2 | puntatore1 |
| 1.3.6.4.12.2.3.4.9.5.5.5.3.4.3.4 | puntatore2 |
| 1.3.6.4.12.2.3.4.7.3.3.3.4.5.6.7 | puntatore3 |
| 1.3.6.4.12.2.3.4.7.1.5.7.3.3.3.4 | puntatore4 |
| ... | ... |

Quindi l'agent può essere diviso in più parti: un proxy che riceve le richieste, verifica i permessi, risolve gli indirizzi e li passa all'agent che si occupa della gestione. L'agent di gestione oltre che trattare le richieste di lettura e scrittura dati si occupa della generazione di TRAP. Naturalmente si possono immaginare più agents di gestione che si occupano ognuno di una porzione dei MOs.
L'organizzazione interna dell'agent, le comunicazioni con il manager e con le risorse fisiche sono puntualizzate nella sezione 2.1.3.

### 2.1.2 I managers

I manager saranno a loro volta implementati in Java, questo rende il sistema uniforme e le comunicazioni agent/manager e manager/manager semplici. Notiamo che, se il sistema deve essere esteso in tutte le sue funzionalità di management di cui alla sezione 1, può essere necessario realizzarne una parte utilizzando altri linguaggi, in questo caso si può definire l'interfacciamento con l'utilizzo di CORBA [CORBA]; sebbene questa soluzione sembri più diretta, provocherebbe un rallentamento e un appesantimento dovuto alla grande quantità di identificatori necessari all'ORB(Object Request Broker) [CORBA]; è quindi più opportuno definirsi delle interfacce (API) adeguate caso per caso.
Le relazioni manager/agent, manager/manager e i collegamenti con le risorse fisiche sono puntualizzate nella sezione 2.1.3.

### 2.1.3 L'interfaccia grafica

L'applicazione di management deve essere definita tenendo conto di tutte le problematiche della gestione e quindi: delle risorse fisiche, di come meglio astrarle individuandone le caratteristiche importanti ai fini che ci interessano ma non deve essere dimenticato il punto di incontro tra l'applicazione e i suoi utilizzatori. L'interfaccia grafica che permette al gestore di interagire con il sistema e di modificarne i parametri nonché di riconoscere gli allarmi in atto è il punto di incontro tra uomo e sistema.

La nostra interfaccia sarà ovviamente programmata in Java, linguaggio che a questo livello esprime meglio le sue potenzialità, permettendoci di definire sia un'applicazione console che riporti tutti gli allarmi e la situazione attuale del sistema, sia un'applicazione internet (applet) che permetta (con le dovute autorizzazioni) di interrogare il sistema riguardo alla localizzazione di utenti e altro.

## 2.2 L'architettura

In questa sezione definiamo più in dettaglio l'architettura di cui la figura 3 mostra i collegamenti tra il manager centrale e uno qualsiasi dei manager locali.
Partendo da questa visione globale del sistema ci addentriamo nel particolare per vedere meglio come sono composte le varie parti del sistema di management.

Partiamo dagli agent figura 4. Come già detto si tratta di script, ma la particolarità sta nell'introduzione di un proxy agent che si occupa di ricevere le richieste che pervengono dal manager locale, di verificare i permessi, e di passarle all'agent opportuno: questo passaggio viene preceduto dalla risoluzione del naiming del MO richiesto. Il proxy gestisce la tabella di naming dalla quale mediante l'identificativo dell'oggetto è in grado di estrarre il puntatore da passare all'agent (utente o comunicazione) che provvederà a gestire la richiesta accedendo al DB opportuno. L'agent utenti realizza i MOs utenti (in realtà ci sono più agent utenti e comunicazioni perché anche grazie all'introduzione del proxy i MOs da gestire sono ancora troppi per un solo agent) e si occupa perciò di rispondere al manager locale riguardo alle richieste pervenute tramite proxy e di inviare le TRAP definite dallo script (come già detto si riferiscono alla creazione di un nuovo MO o alla sua cancellazione e al superamento di alcune soglie come il numero di utenti per area). L'agent utenti, per far fronte alle richieste, accede al DB VRL per leggere i dati dell'utente e al DB locale per inserire e prelevare informazioni storiche (es: ora di deregistrazione per la TRAP).

L'agent delle comunicazioni ha le stesse funzioni del precedente ma in più si occupa di inviare le informazioni per la tariffazione (numero chiamante,

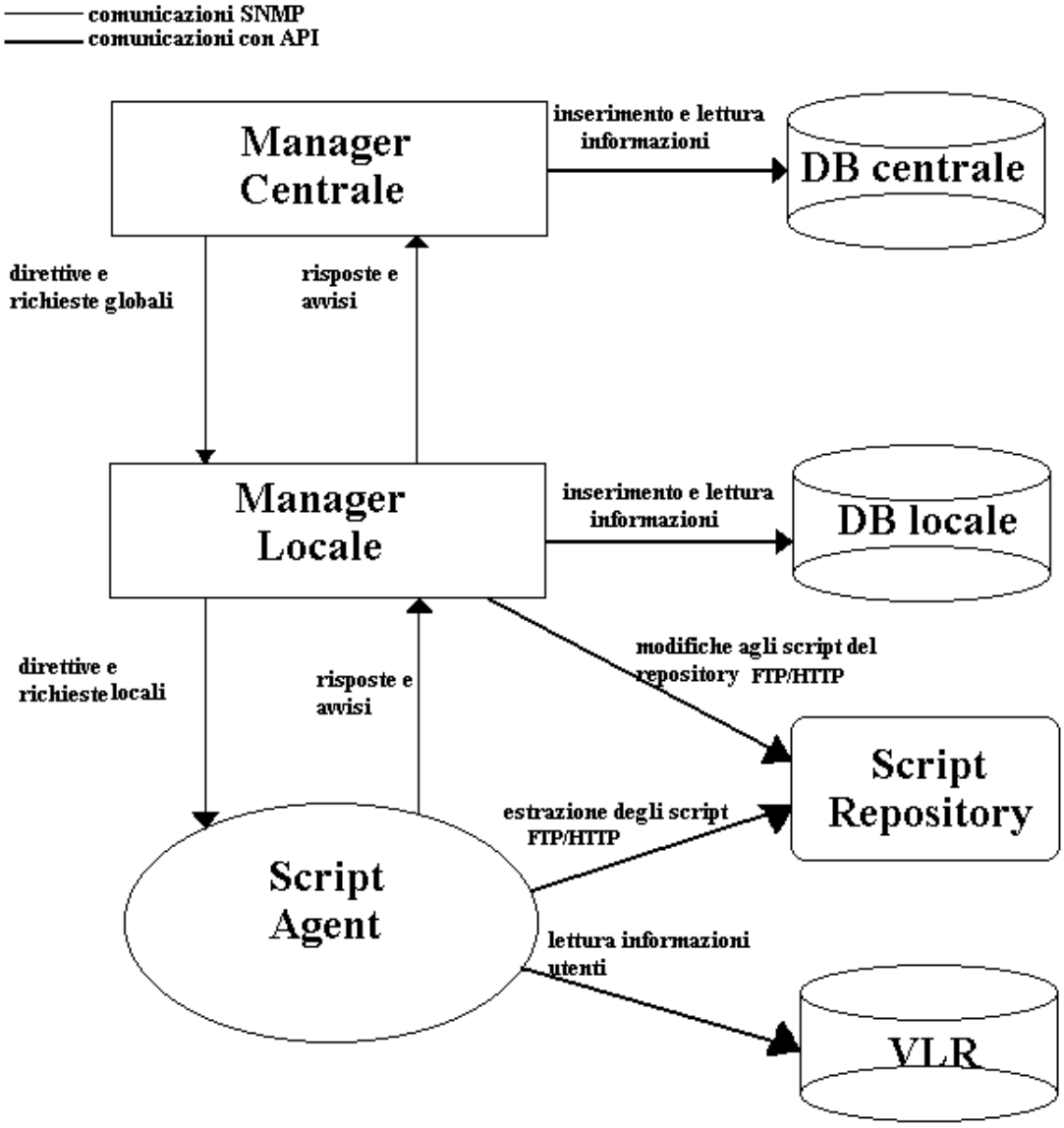


Figura 3: Comunicazioni tra: manager centrale, locale e agent.

numero chiamato, ora di inizio e ora di fine conversazione) al sistema di billing.

Ho introdotto anche un gestore per gli script, riceve le segnalazioni di modifica degli script dal manager centrale e provvede a riconfigurare gli agent eseguendo i nuovi script recuperati dallo script repository.

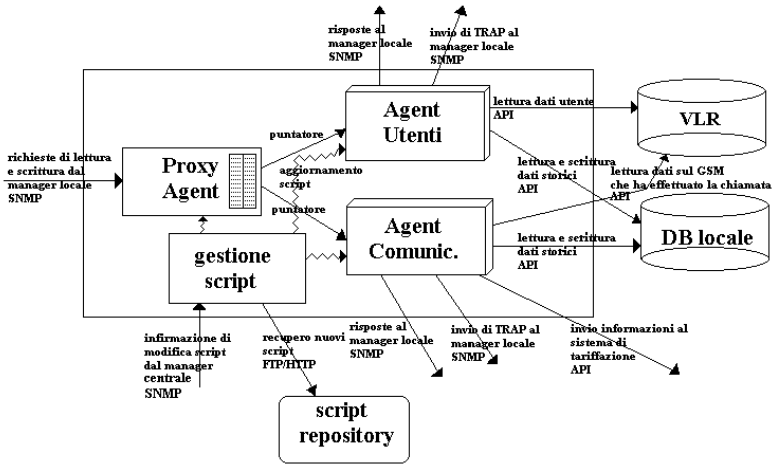


Figura 4: Agent in dettaglio.

I manager locali (figura 5) si occupano del management specifico della loro zona, possono perciò avere priorità e direttive diverse ma la loro struttura è la stessa.

Ho previsto un analizzatore di TRAP che, dopo aver ricevuto una TRAP dall'agent, la processa per verificarne la veridicità nonché l'utilità prima di inviarla al gestore della console. Le informazioni raccolte durante il processo di analisi vengono inserite nel DB locale in previsione di un uso futuro (es: per raffronti con nuovi dati o per la realizzazione di report per le scelte di management). Inoltre le TRAP ritenute di interesse globale vengono riportate al livello superiore (al manager centrale).

Il gestore della console riceve le TRAP filtrate e provvede ad informare l'applicazione di console che si occuperà di visualizzarle sulla sua interfaccia grafica.

Dalla console possono pervenire richieste di lettura e modifica di valori riferiti agli utenti e alle comunicazioni, in questo caso il gestore provvede dopo le dovute verifiche ad inoltrarle all'agent (in realtà al proxy).

il gestore della console si occupa, inoltre, delle operazioni di polling. Tali operazioni sono necessarie in quanto usando SNMP non si è in grado di sapere se un agent è attivo se non chiedendoglielo (se un agent continua a mandare TRAP sappiamo che é attivo ma si spreca un sacco di banda).

Vediamo infine il manager centrale (figura 6). La parte forse più interessante del manager centrale è quella di interfacciamento con l'applet Java destinata alle richieste da web, a tale scopo ho previsto un gestore delle

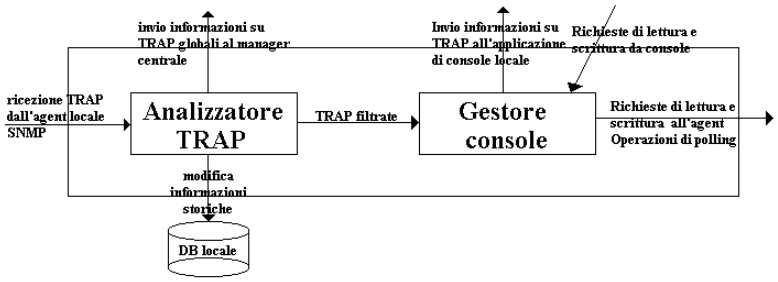


Figura 5: Manager locale in dettaglio.

richieste dall'applet. Tale gestore può essere realizzato come un server che riceve le richieste dall'applet attraverso una interfaccia Java, il che garantisce la sicurezza nella gestione delle password, oppure come un processo CGI che gestisce le richieste e propone i risultati in un formato predefinito, limitando di molto la sicurezza del sistema.

La scelta tra i due casi proposti va riferita al tipo di informazioni che devono essere inviate all'applet e al grado di sicurezza che ci interessa.

Notiamo che l'introduzione del DB centrale a questo livello permette di ricercare le informazioni richieste prima di tutto nel DB centrale e solo in caso di fallimento si dovrà inoltrare una richiesta ai manager locali (in questo modo si risparmia banda).

Anche a questo livello troviamo un gestore della console che come al solito riceve le TRAP filtrate dall'analizzatore (che ha la funzione di elaborarle per estrarne le più significative e di riportare le informazioni storiche nel DB) e le invia all'applicazione di console che le visualizza sull'interfaccia grafica.

Una parte del manager centrale si occupa della configurazione, è qui che vengono analizzate le informazioni storiche e che vengono prese decisioni sulla modifica degli script agent (naturalmente le decisioni prese in automatico dal sistema riguardano solo gli aspetti più semplici del management mentre le decisioni più importanti possono essere prese solo dal manager utente).

In un primo momento (vedi figura 3) il compito di modificare gli script era stato delegato ai manager locali ma poi ho pensato che solo il manager centrale, avendo una visione di tutto il sistema, può prendere decisioni di questo tipo.

# 3 Commenti finali

In questo proggetto abbiamo affrontato il problema della progettazione architetturale solo da un punto di vista generale. Restano da fare alcune scelte importanti di progettazione di dettaglio, dipendenti dal sistema e dal livello di sicurezza che si vuole raggiungere. Alcune scelte o linee di condotta potrebbero essere addirittura riviste durante la progettazione di dettaglio, in particolar modo mi riferisco a tutto ciò che prevede l'uso esplicito di Java,

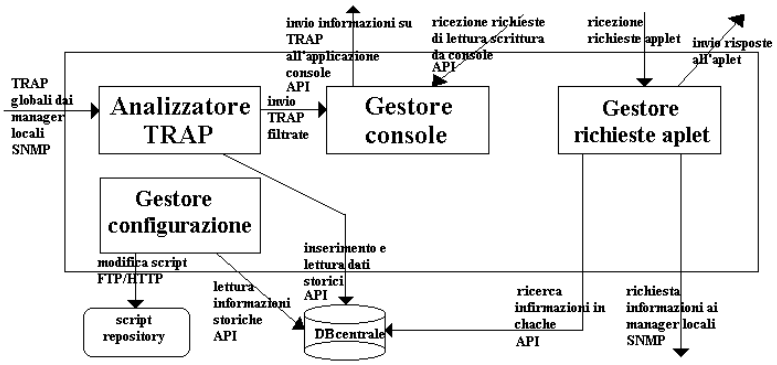


Figura 6: Manager centrale in dettaglio.

potrebbe rendersi utile in fase implementativa rivolgersi ad un altro linguaggio o addirittura prevederne più di uno a seconda della parte da realizzare (questa seconda ipotesi è più verosimile se, come in questo caso, alcune parti del sistema sono state realizzate in precedenza). Infine non è stato tratto approfonditamente il problema dell'interfacciamento tra le varie parti dell'architettura, riconoscendo solo le comunicazioni tramite SNMP e indicando con il termine API tutte le interfacce da definire (escluso alcune che richiedono l'utilizzo di HTTP o FTP).

## Riferimenti bibliografici

[Management] J. Schönwälder e L. Deri: Sistemi di Elaborazione dell'Informazione: Gestione di Rete
Lucidi del corso.
Anno Accademico 2000/ 2001
Disponibile all'indirizzo:
http://www.luca.ntop.org/Teaching/CorsoNwMgnt-v1.0.pdf

[GSM] Javier Gonzálvez Sempere:
An overview of the GSM system
Disponibile all'indirizzo: www.comms.eee.strath.ac.uk/∼gozalvez/

[Java] Il sito Java della Sun: http://java.sun.com

[CORBA] http://www.cs.wustl.edu/∼schmidt/tutorials-corba.html
http://www.cs.indiana.edu/hyplan/kksiazek/tuto.html
http://www.cs.umbc.edu/∼thurston/cbatop.htm
http://armyant.ee.vt.edu/CORBAtutorial/

[TRAP] Francescco Galli: Architettura per la correlazione di trap SNMP provenienti da probe distribuiti.
Disponibile all'indirizzo: http://www.luca.ntop.org/Teaching/FrancescoGalli.pdf