ARCHITETTURA PER IL MONITORAGGIO DI SERVIZI DISTRIBUITI: IL CASO AKAMAI

Barone Luca
Battistini Alessio

CORSO DI SISTEMI PER L'ELABORAZIONE DELL'INFORMAZIONE: GESTIONE DI RETE

AA 2000/2001

Sommario

1. INTRODUZIONE

1.1 AKAMAI

Akamai è un'azienda che fornisce servizi di rete, con lo scopo di ottimizzare i tempi di richiesta e risposta per architetture client – server. Akamai ha a disposizione 9700 server distribuiti in 650 reti in più di 56 paesi nel mondo. Il funzionamento di Akamai sta nel redirigere le richieste del client verso un server-akamai piu vicino ad esso utilizzando algoritmi di routing per calcolare il cammino ottimo, questi algoritmi cercano un server akamai che sia più vicino possibile al client e che non sia troppo carico, in modo che i tempi di risposta del server al client siano i più bassi possibili.
Come si vede dalla figura 1, il client fa una richiesta di pagina ad un web server e quest'ultimo consegna la prima pagina al client con un tag che servirà a redirigere le prossime richieste al server akamai più vicino o comunque ad uno vicino che non sia carico.

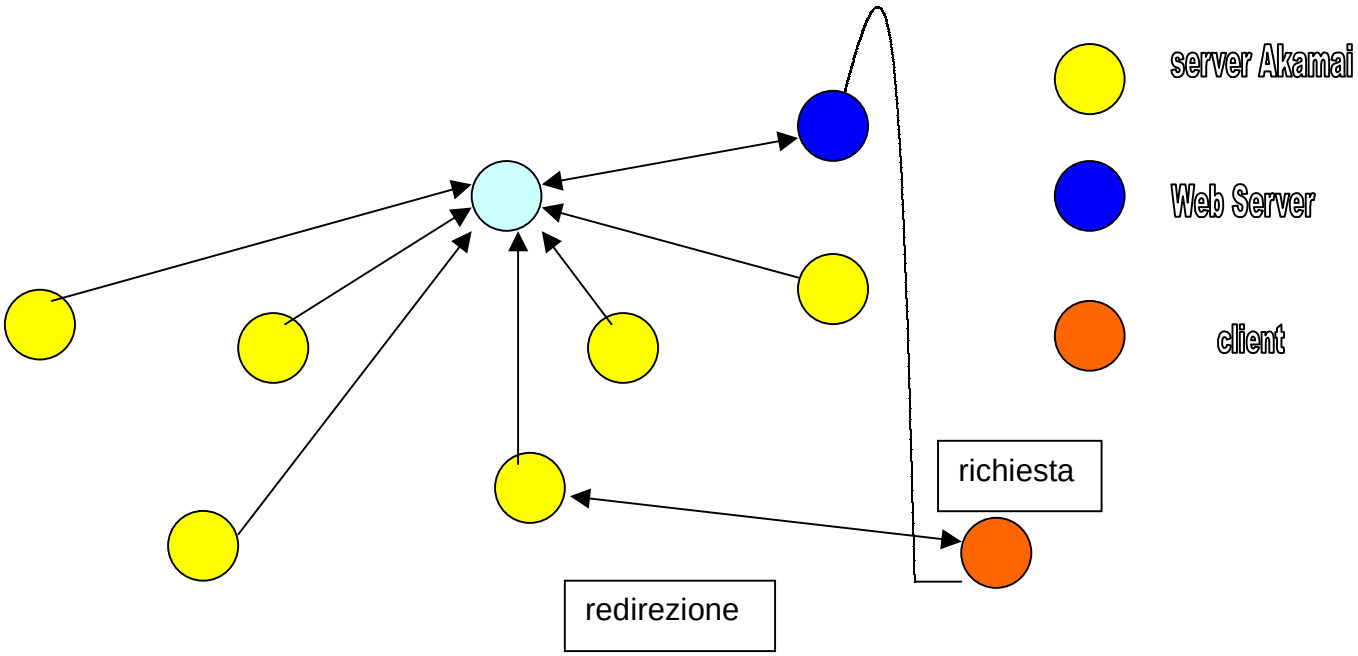

Fig.1: Akamai

1.2 SCOPO DEL PROGETTO

Lo scopo del progetto e' quello di fornire un'architettura per la gestione di un servizio distribuito come quello fornito da Akamai. La gestione consiste nel monitorare i vari server Akamai e notificare eventuali problemi alla console di management.
Le notifiche che appaiono sulla console devono rilevare problemi di una certa importanza sulla performance dei server Akamai.
Quello che dobbiamo controllare e': lo stato dei server se e' UP/DOWN, il numero di richieste che attualmente soddisfa, in particolare se il server e' sovraccarico oppure scarico e la velocità di trasmissione del server.
Per questo motivo abbiamo proposto un'architettura che si adatti all'elevato numero di server e che allo stesso tempo sia in grado di selezionare, cioè di far arrivare alla console di management le notifiche che riguardano problemi effettivamente importanti. Tramite questo monitoraggio l'azienda Akamai deve capire se c'e' bisogno di intervenire nei vari territori per allocare o disallocare server Akamai allo scopo di migliorare il servizio.

2. ARCHITETTURA GENERALE

2.1 MANAGEMENT BY DELEGATION

L'architettura sviluppata è una struttura gerarchica del sistema di gestione precisamente una architettura basata sul paradigma Management by Delegation , cioè cerchiamo di portare le applicazioni di management più vicino possibile ai server Akamai in modo da avere meno traffico per la gestione e meno carico di lavoro sulle macchine a più alto livello.
L'architettura prevede un agent installato su ogni macchina Akamai, questi agent hanno il compito di controllare le variabili del MIB e di notificare ai manager locali (trap SNMP) eventuali malfunzionamenti del server.
Ai manager locali e a quelli di livello superiore spetta il compito di capire quali siano le informazioni rilevanti e quindi di trasmetterle ai livelli superiori fino ad arrivare alla console di management.
Il compito di definire le informazioni importanti spetta all'amministratore, che in base al funzionamento dei server nel tempo deve fissare dei parametri

sui quali gli agent e i manager si basano per emettere notifiche. Per questo motivo, è importante avere a disposizione dei dati storici in base ai quali è giusto fare delle considerazioni, questi dati vengono richiesti espressamente e periodicamente (GET SNMP) dai manager che devono avere una visione globale dello stato dei server che gestiscono.

Architettura del sistema

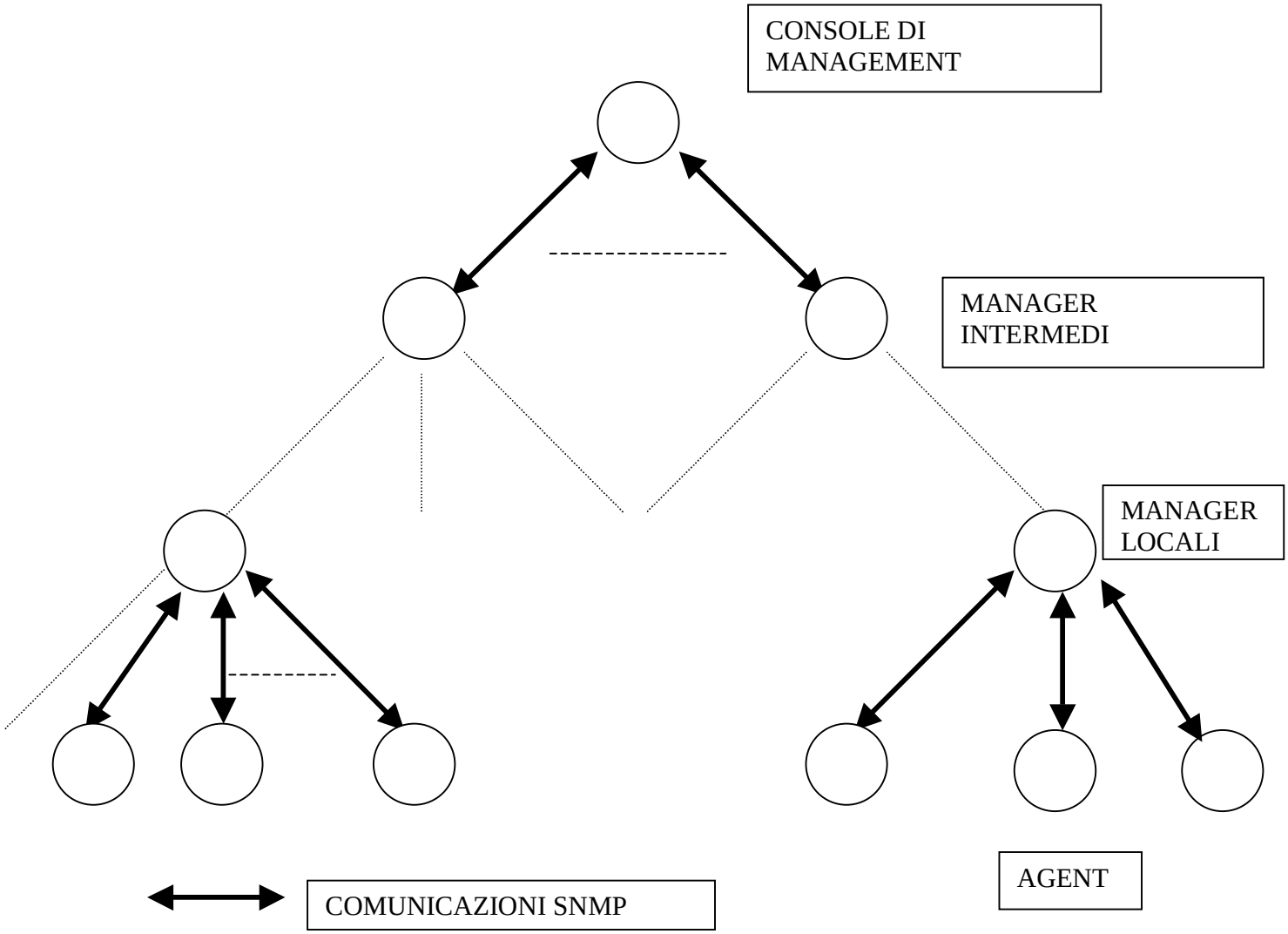

Fig. 2: architettura del sistema

2.2 GLI AGENT

L'agent deve controllare le istanze del MIB che descrivono la performance del server Akamai. In particolare controlla nel MIB il valore delle variabili; queste variabili sono: il numero di utenti connessi e la velocità di trasmissione, confronta il suddetto valore delle variabili con i valori di soglia che ha a disposizione e nel caso il confronto dia un risultato non aspettato, cioè, se il valore di queste variabili è più alto della soglia massima oppure più basso della soglia minima deve inviare delle notifiche (trap SNMP) al manager locale.
I valori di soglia devono essere periodicamente controllati ed eventualmente modificati, infatti tali valori possono cambiare nel tempo, in quanto può cambiare la distribuzione dei server Akamai (ad esempio ci può essere la necessità di installare altri server Akamai nelle zone dove i server sono più carichi, oppure di sostituire alcuni server mettendone di più potenti o addirittura di togliere quei server che sono sottoutilizzati). Inoltre l'agent deve inviare trap appena il valore della variabile supera la soglia e non tutte le volte che il valore cambia ma resta sopra tale limite, (vedi fig.3) questo serve per non tempestare il manager locale con continue ed inutili trap (più avanti definiremo quattro trap outmaxAccess, outminAccess, inmaxAccess e inminAccess che appunto vengono emesse nel caso il numero di accessi al server superano (out) e rientrano (in) nelle soglie stabilite).

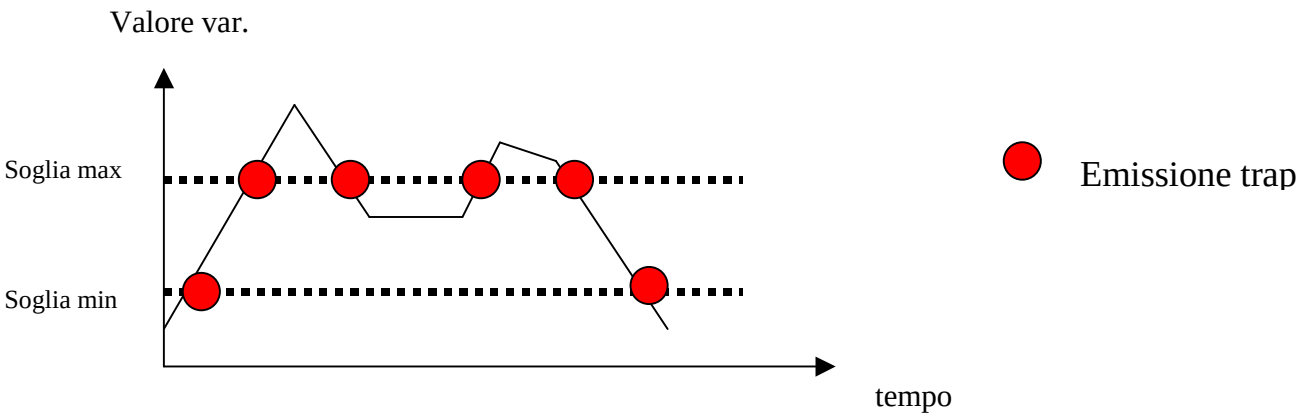

Fig.3: emissione delle trap

Inoltre l'agent deve informare il manager locale ogni qualvolta si reinizializza cioè ogni qualvolta torna attivo (UP). Questa informazione serve al manager per capire se il server è per qualche motivo caduto. A tal proposito facciamo notare che questo tipo di trap non sono sufficienti al manager, infatti questi verrebbe a conoscenza della caduta del server solo quando il server torna attivo ma non saprebbe per quanto tempo è rimasto "down", è per questo che adottiamo una strategia di polling adattata alle trap ricevute, cioè se l'agent per un periodo di tempo, che l'amministratore dei servizi Akamai deve stimare, non invia trap, il manager va ad interrogare con una operazione di polling l'agent per vedere se è attivo oppure no.
Possiamo anche dotare l'agent di script MIB che sono mandati in esecuzione ogni qualvolta il manager lo richieda. Questi script vanno a controllare le risorse del server Akamai per avere un quadro completo della situazione del server. Questi script possono ad esempio andare a controllare l'occupazione del disco dei server oppure eseguire operazioni su più variabili del MIB, infatti sappiamo che SNMP può richiedere letture e scritture di variabili ma non eseguire altre operazioni più complesse su di esse.

2.3 I MANAGER

Il manager locale è delegato a controllare gli agent che gestisce, in particolare deve controllare che gli agent e quindi i server Akamai siano attivi (UP) attuando una politica intelligente di polling cioè adattando la strategia di polling alle trap ricevute.
Inoltre il manager locale ad intervalli di tempo regolari richiede all'agent di fornirgli (GET SNMP) lo stato delle risorse critiche del sistema in modo da memorizzare tali dati nel database per avere delle statistiche utili e per controllare l'efficienza del server.
Tali Get Snmp come abbiamo detto possono mandare in esecuzione sull'agent degli script che vengono definiti dai manager e vengono memorizzati nello Script Repository.
I manager, a qualsiasi livello dell'architettura si trovino, devono essere implementati in maniera intelligente cioè in modo da selezionare le trap che gli arrivano e di non tempestare la console di management o i manager che gli stanno sopra nell'architettura con notifiche "poco utili", anche perché, stiamo monitorando un elevato numero di server (9700). Più avanti nel documento proveremo a spiegare quali sono i casi in cui è necessario mandare una notifica alla console di management centrale.

Le comunicazioni tra manager di basso livello con quelli che gli stanno sopra avviene tramite SNMPv1 e quindi tutte le notifiche sono di fatto delle trap.

2.4 IMPLEMENTAZIONE DEGLI AGENT E MANAGER

I manager e gli agent comunicano tramite SNMPv1 in particolare utilizziamo Get (per richieste manager-agent e manager-manager) e delle Trap (per notifiche agent-manager e manager-manager), per quanto riguarda la loro implementazione supponiamo di utilizzare Java che mette a disposizione una serie di librerie per la gestione di rete, inoltre possiamo usare JDBC che implementa interfacce per database SQL da noi utilizzati per memorizzare i dati statistici. Tramite queste interfacce e' possibile collegarsi con estrema semplicità a qualsiasi database locale oppure (e qui sta la caratteristica principale di JDBC) a database remoti indipendentemente dal DBMS utilizzato. Abbiamo utilizzato SQL per la creazione di database in quanto e' largamente utilizzato e facile da accedere attraverso librerie Java.
Mentre l'interfaccia tra SNMP e Java avviene tramite delle interfacce AdventNet. AdventNet è un'azienda che ha realizzato uno stack SNMP scritto in Java.

Manager e agent

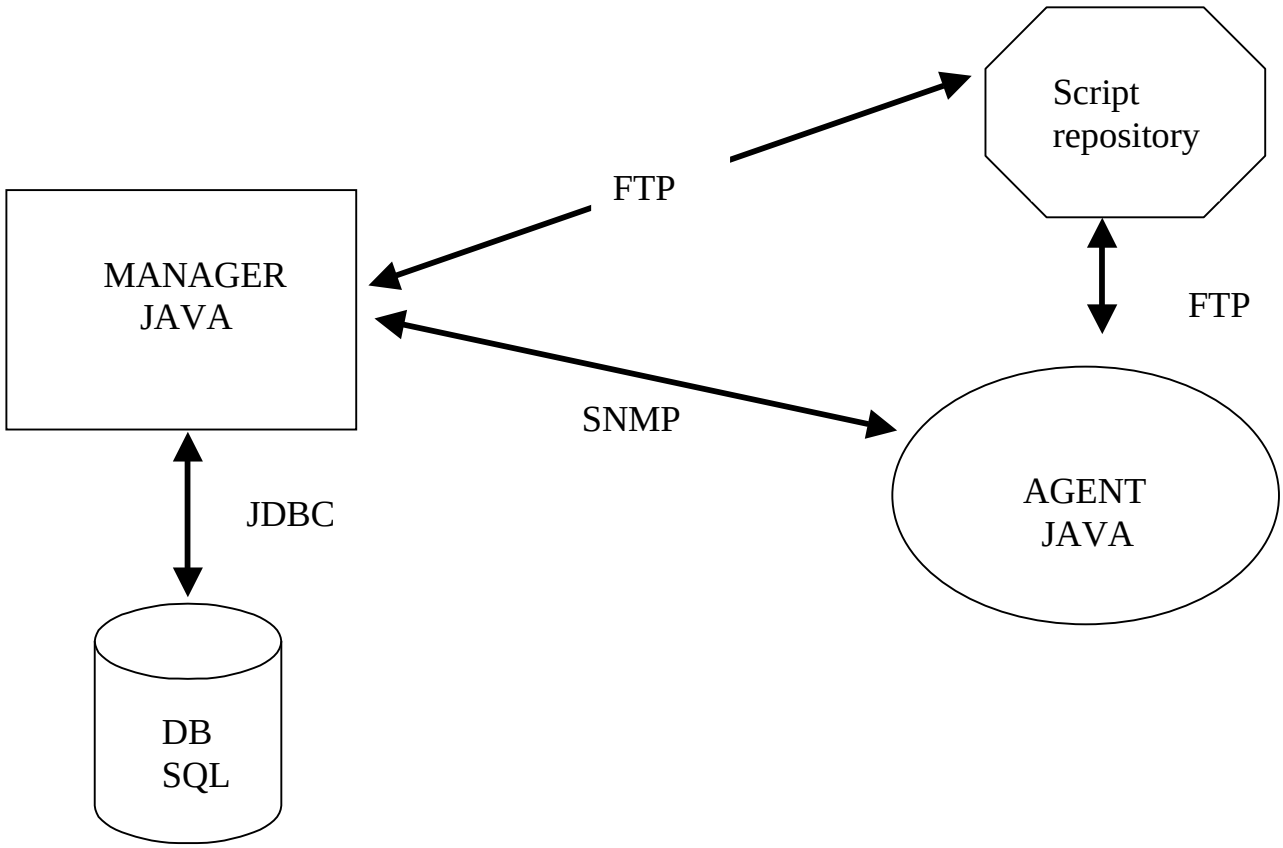

Fig.4: manager e agent

2.5 NOTIFICHE ALLA CONSOLE DI MANAGEMENT

I manager intermedi devono filtrare adeguatamente le notifiche che gli giungono. Questo compito è più gravoso per i manager locali in quanto gli agent emettono notifiche ogni qualvolta si verifica un superamento di un valore di soglia, ma questo fatto in certe situazioni o per brevi periodi di tempo può essere poco rilevante.
Infatti potrebbe accadere che per un blackout il server cada per un periodo di tempo non troppo grande, il manager locale si accorgerà di questo, ma non

avviserà i manager superiori, almeno che, questo fatto non si verifichi frequentemente. Le stesse considerazioni valgono per lo stato (sovraccarico/scarico, lento/veloce....) del server che in certi periodi dell'anno e in certi momenti della giornata possono non essere quelli soliti ma tuttavia possono essere aspettati, infatti il 15 agosto si può immaginare che le richieste ai server Akamai italiani siano minori di quelle che si hanno in un normale giorno lavorativo e quindi non notificati alla console.
I casi che abbiamo analizzato sono i seguenti: nel caso in cui il manager locale si accorge tramite una operazione di polling che l'agent di un server Akamai è giù e quindi anche il server stesso, il manager, oltre a registrare l'evento nel proprio database, si mette in attesa per un periodo di tempo (che l'amministratore del servizio deve stimare) di una trap da parte dell'agent (caduto) che notifica la ripartenza del server, se ciò non avviene il manager verifica che il server sia ancora giù (infatti la trap di reinizializzazione spedita dall'agent può andare persa, non dobbiamo dimenticarci infatti che SNMP si basa su UDP) e se così è, invia una trap al manager soprastante nell'architettura.
Nel caso in cui il server Akamai risulta essere sovraccarico/scarico cioè nel caso in cui il valore di num_access nel MIB è sopra/sotto i valori di soglia stabiliti allora l'agent invierà una trap (outmaxAccess/outminAccess nel MIB). Nel caso in cui il manager locale non riceva entro un tempo (che l'amministratore dei servizi akamai deve stimare) una trap inmaxAccess/inminAccess (vedi MIB) allora invierà una trap al manager soprastante. A questo punto saranno i manager intermedi che avendo una visione più ampia dello stato dei server decideranno se inoltrare trap verso l'alto oppure ignorare tale notifica, se non addirittura impostare nuovi valori di soglia tramite delle operazioni di SET SNMP.

3.DEFINIZIONE DEL MIB

Questo e' il Mib da noi utilizzato per controllare la performance del server Akamai :

Struttura del MIB:

```
AKAMAI-MIB DEFINITION ::= BEGIN

IMPORTS
      MODULE-IDENTITY,  OBJECT-TYPE,   Mib-2, counter64
                FROM SNMPv2-SMI
      MODULE-COMPLIANCE, OBJECT-GROUP
                FROM SNMPv2-CONF



akamai-MIB  MODULE-IDENTITY
      LAST-UPDATED "……"
      CONTACT-INFO "……."
      ORGANIZATION "……."
      DESCRIPTION    '’……."
.
.
.
   ..= { private 65}


akamaiService  OBJECT-IDENTIFIER ::= { akamai-MIB  1}

akamaiConformance OBJECT-IDENTIFIER : :={ akamai-MIB 2}


num_access  OBJECT-TYPE
      SYNTAX          counter64
      MAX-ACCESS  read-only
      STATUS           current
      DESCRIPTION
                        " numero di accessi al web server akamai"
      :: =  {akamaiService  1 }




speed_server  OBJECT-TYPE
```

```
        SYNTAX          counter64
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
                        "numero di pacchetti  spediti al secondo"
      ::=  {akamaiService 2 }



stato OBJECT-TYPE
        SYNTAX          Integer
        MAX-ACCESS      read-only
        STATUS          current
        DESCRIPTION
                        "indica lo stato del server Akamai, prende i valori 1 se up 0 se
                        down"
        ::={akamaiService 3}



-- Trap

statoServer TRAP-TYPE
        ENTERPRISE      Akamai
        VARIABLES       { stato}
        DESCRIPTION
                         "trap che viene inviata quando il server torna attivo se stato = 1
                         oppure quando e caduto se stato =0 "
        ::=1


outminAccess TRAP-TYPE
        ENTERPRISE      Akamai
        VARIABLES       {num_access}
        DESCRIPTION
                        "trap che viene inviata quando il valore di num_access è sotto la
                        soglia minima "
  ::=2
```

```
inminAccess TRAP-TYPE
      ENTERPRISE    Akamai
      VARIABLES     {num_access}
      DESCRIPTION
                    "trap che viene inviata quando il valore di num_access torna
                    sopra la soglia minima "
  ::=3


outmaxAccess TRAP-TYPE
      ENTERPRISE    Akamai
      VARIABLES     {num_access}
      DESCRIPTION
                     "trap che viene inviata quando il valore di num_access è sopra
                     la soglia massima"
     ::=4


inmaxAccess TRAP-TYPE
      ENTERPRISE    Akamai
      VARIABLES     {num_access}
      DESCRIPTION
                     "trap che viene inviata quando il valore di num_access torna
                     sotto la soglia massima"
     ::=5



--conformance information

a_mibcompliances OBJECT-IDENTIFIER
         ::={akamaiConformance 1}


a_servergroups OBJECT-IDENTIFIER
         ::={akamaiConformance 2}
```

```
a_compliance MODULE-COMPLIANCE
      STATUS current
      DESCRIPTION
                  "le dichiarazioni per l'entita' snmp che akamai-MIB implementa"
       MODULE –questo modulo
       MANDATORY-GROUPS {a_group}
       ::={a_mibcompliances 1}




a_group OBJECT-GROUP
      OBJECT {num_access, speed_server, stato}
      STATUS current
      DESCRIPTION
                        "collezione degli oggetti utilizzati in questo mib"

      ::= {a_servergroups 1}


END
```

Le trap definite vengono inviate dall'agent al manager quando i valori delle variabili non rispettano le soglie fissate. Un caso a parte è la trap statoServer che viene inviata dall'agent al manager quando il server si reinizializza, oppure viene inviata dal manager al manager soprastante nell'architettura, quando ci si accorge con una operazione di polling che il server Akamai è nello stato down.

4. COMMENTI FINALI

Lo scopo di questo progetto è quello di fornire una architettura per la gestione di servizi distribuiti, ovviamente restano da fare molte scelte importanti di progettazione, che solo chi conosce bene il funzionamento del servizio può fare. Non possiamo infatti entrare nel dettaglio definendo i valori di soglia e i tempi di attesa per effettuare il polling e per l'invio di trap, che come abbiamo scritto sono valori che l'amministratore di progetto deve definire e modificare nel tempo.
Inoltre non abbiamo dato tanto peso al fatto che SNMP si basa su UDP e che quindi i messaggi possono essere persi, a tal proposito, per esempio, potrebbe essere inserito un database utilizzabile dall'agent per memorizzare tutte le trap inviate.
Abbiamo cercato di proporre un'architettura semplice che non dia troppo carico di lavoro agli agent e che distribuisca il lavoro dei manager sui vari livelli dell'albero, tuttavia in fase di progettazione alcune scelte potrebbero essere riviste e modificate, per esempio per diminuire la complessità dell'agent potrebbe essere omesso l'utilizzo di script che non è di fondamentale importanza nella nostra architettura.

5. REFERENZE

Per lo sviluppo di questo progetto abbiamo consultato i seguenti documenti:

J. Schönwälder e Luca Deri : Sistemi per l'elaborazione dell'informazione: gestione di rete (lucidi del corso A.A 2000/2001)
http://luca.ntop.org/Teaching/CorsoNwMgnt-v1.0.pdf

Il sito della Sun: http://java.sun.com

Paola Filippini: MIB SNMP per la gestione di un server www
Disponibile all'indirizzo : http://luca.ntop.org/Teaching/PaolaFilippini.pdf

De Col Daniela : Architettura distribuita per la gestione di reti GSM.
Disponibile all'indirizzo : http://luca.ntop.org/Teaching/DanielaDeCol.pdf

Il sito AdventNet: http://www.adventnet.com