I Mobile Agents nel Network Managment

Roberto Morini

1 Introduzione

Gli approcci tradizionali al problema del network managment (la gestione delle reti) sono caratterizzati da architetture gerarchiche o centralizzate e sono basati sul paradigma client/server; queste scelte progettuali hanno ben note limitazioni. Di fronte alla costante crescita della complessità delle reti e delle telecomunicazioni si è portato avanti il bisogno di studiare tecnologie per la gestione delle reti distribuite, con l'obiettivo della flessibilità, della scalabilità e dell'apertura alle estensioni.Uno dei più promettenti approcci è la tecnologia ad agenti intelligenti, che non solo supporta applicazioni distribuite ma anche la mobilità del codice. Tuttavia, nonostante gli sforzi della ricerca in questo campo, non c'è molto di queste tecniche nel mondo della gestione delle reti. Uno dei motivi potrebbe essere che le piattaforme ad agenti disponibili sono molto generali nella loro concezione e non affrontano i particolari problemi di questo ambito. Più avanti saranno descritte alcune architetture e piattaforme pensate proprio per risolvere questo tipo di problema.

Le applicazioni per la gestione delle reti (network managment) sono basate principalmente su 2 protocolli: SNMP largamente usato nelle reti IP (che verrà illustrato più nel dettaglio in seguito) e CMIP per reti di telecomunicazioni. Questi protocolli sono basati su architetture client / server statiche, centralizzate e non scalabili, in cui alcune entità centrali processano grandi quantità di dati grezzi raccolti da ogni elemento di rete. La necessità di applicazioni di gestione più scalabili e flessibili sta portando ad un intensa ricerca per una maggiore decentralizzazione, con approcci come la Managment By Delegation, CORBA, Web-based management, agenti intelligenti, Active Networks e, più recentemente, Mobile Agent Technology.

Un Mobile Agent può essere descritto come un piccolo programma software che è in grado di migrare tra gli hosting computer durante la sua esecuzione attraversando tutta la rete. Con questo, il task da compiere può essere dinamicamente distribuito attraverso la rete e posto più vicino alla gestione dei dati. Rispetto alle classiche soluzioni client-server, la Mobile-Agent technology riduce il traffico di rete e aumenta la scalabilità, la flessibilità e la robustezza. La Mobile Agent Technology è stata applicata a diversi settori, come mobile computing, e-commerce, servizi internet, ricerca di informazioni, e la gestione della rete.

La gestione della rete è spesso considerata come uno dei campi di applicazioni con maggior potenziale per la tecnologia dei Mobile Agent. Tuttavia, le applicazioni e le architetture già sviluppate e utilizzate impongono un notevole rallentamento alla diffusione di nuove soluzioni per la gestione della rete. Per questa ragione, le infrastrutture per i Mobile Agents dovranno integrarsi sen-

za sforzo, con quadri di gestione già esistenti nel patrimonio tecnologico come l'architettura diffusa SNMP.

Il software per il Network Management è attualmente dominato da sistemi basati su tecnologie client/server. Nella maggior parte dei casi, questo approccio produce soluzioni monolitiche, non scalabili e poco flessibili. In genere, secondo questa visione, ci sono diversi dispositivi distribuiti in tutta la rete, in ogni dispositivo è installato un server statico (o agente) che è responsabile della raccolta dei dati grezzi riguardanti il proprio dispositivo locale, con poco o nessun pre-trattamento dei dati. Nell'host manager c'è un processo client che interagisce con tutti i server nei dispositivi di rete, raccoglie le informazioni da essi e fornisce le informazioni necessarie all'utente finale. Questi server (o agenti) sono statici e perciò sono molto difficili da aggiornare o da personalizzare.

In questo modo si crea un collo di bottiglia a livello dell'host manager a causa della natura centralizzata del sistema, inoltre alcune delle applicazioni sono molto inefficienti per quanto riguarda l'uso della larghezza di banda della rete, poiché la maggior parte dei dati deve essere inviata dai dispositivi di rete all'host manager per essere trasformata. Questi dati grezzi sono prodotti dai server in grandi quantità e hanno formati diversi a seconda del client o della versione del software installato.

Dato questo scenario, l'uso di agenti mobili per la raccolta dei dati può fornire dei potenziali benefici. Prima di tutto, i Mobile Agents consentono una prima trasformazione dei dati direttamente alla sorgente, così da aumentare la scalabilità e ridurre la congestione e il traffico della rete. La robustezza dell'applicazione è migliorata attraverso l'uso di questi agenti autonomi, i Mobile Agent, che permettono anhe di risolvere il problema di diversi formati di dati e di configurazioni eterogenee dei vari dispositivi che compongono la rete. Un altro importante aspetto è quello dell'aggiornamento dell'applicazione, che, grazie all'uso dei Mobile Agent, richiede solamente la codifica di nuovi agenti e la distribuzione automatica di essi sulla rete gestita. In questo modo, l'estensione delle funzionalità o l'installazione di nuove versioni del software diventa più flessibile e senza sforzo.

Per questi motivi i Mobile Agents sono stati proposti come soluzione per la gestione distribuita delle reti.

2 I Mobile-Agents

L'approccio centralizzato nella gestione delle reti, com'è noto, presenta gravi limitazioni riguardo a efficienza e scalabilità: il processo di raccolta dei dati e l'analisi comportano tipicamente massicci trasferimenti di dati ponendo a dura prova il throughput di rete formando un collo di bottiglia sul manager host a causa del forte carico di dati da processare. Tutti questi problemi suggeriscono la distribuzione della capacità di gestione come un approccio razionale per superare i limiti del Network Managment centralizzato.

Il primo approccio proposto è noto come RMON (Remote MONitoring) che introduce un primo grado di decentralizzazione, questa applicazione è in grado di monitorare il traffico e l'attività nella rete. In termini di attività di ricerca recenti, Management by Delegation rappresenta un impegno chiaro verso il decentramento della logica di gestione, l'approccio iniziale era quello di scaricare script di gestione che sono stati compilati ed eseguiti sul lato agente, questo mo-

do di decentralizzare i compiti è di per se statico (agenti statici) in quanto ogni agente risiede su un host. Tuttavia, l'avvento di Java ha reso possibile prendere in considerazione l'idea di progettare agenti mobili indipendenti dall'host in cui risiedono e di conseguenza in grado di spostarsi da un dispositivo all'altro.

L'affermazione di Java fornisce uno strumento platform indipendent che si presta molto bene all'implementazione e allo studio dell'idea di distribuzione di gestione, proseguita con soluzioni che sfruttano i Mobile Agent. Attaverso i Mobile Agent viene fornito un potente paradigma di interazione software che consente al codice di migrare tra gli host per l'esecuzione remota. Il problema della trasmissione dati può essere affrontato, delegando il lavoro dai manager agli agenti, che sono in grado di filtrare i dati da processare a livello locale, senza la necessità di trasmissione a un gestore centrale. Questa capacità ha attirato molta attenzione sulla tecnologia dei Mobile Agent, con diversi Mobile Agent Framework proposti per le applicazioni di network managment.

E' molto importante capire la differenza tra Mobile Code e Mobile Agent, poichè la linea di demarcazione tra le due tecnologie è una fonte costante di dibattito. I Mobile Agent tendono ad avere un grado di intelligenza elevato e di solito hanno un livello di autonomia che li rende in grado di migrare da nodo a nodo e di comunicare con gli altri agenti, mentre il Mobile Code ha capacità più limitate. In ogni caso i Mobile Agent sembrano essere molto adatti ad applicazioni di gestione di rete grazie alla loro flessibilità e a tutte le caratteristiche proprie della programmazione ad agenti distribuiti.

2.1 Aglet: un primo modello verso i Mobile Agent

IBM Aglets Workbench (Aglets) rispecchia il modello delle applet in Java per la mobilità del codice e l'implementazione di Mobile Agent. L'obiettivo era quello di portare la caratteristica della mobilità all'applet, infatti la parola Aglet significa proprio unione tra AGenti e appLET. Questo modello è stato progettato per definire un insieme di astrazioni e di comportamenti necessari per poter sfruttare la tecnologia dei Mobile Agent in reti aperte come Internet. Le astrazioni chiave sono:

Aglet. Un aglet è un oggetto Java mobile che visita gli host abilitati in una rete. E' autonomo, dal momento che dopo essere arrivato sull'host, gira su un proprio thread di esecuzione, e reattivo, per la sua capacità di rispondere ai messaggi in arrivo.

Proxy. Un proxy è un rappresentante di un aglet. Opera come uno scudo che protegge l'aglet dall'accesso diretto ai suoi metodi pubblici e può nascondere la sua reale posizione.

Contesto. Un contesto è lo spazio di lavoro dell'aglet. Si tratta di un oggetto stazionario, che fornisce un mezzo per il mantenimento e la gestione in esecuzione dell'aglet in un ambiente uniforme in cui il sistema host è protetto da eventuali aglet dannosi. Un nodo in una rete di computer può eseguire più server e ogni server può ospitare diversi contesti, i contesti hanno un nome e quindi possono essere localizzati.

Messaggio. Un messaggio è un oggetto scambiato tra aglet, sono permessi sia messaggi sincroni che asincroni. Lo scambio di messaggi può essere utilizzato dagli aglet per collaborare e per scambiarsi informazioni.

Risposta Futura. La risposta futura (Future Reply) è usata nel messaggio asincrono, come un handler che riceverà un risultato più tardi in modo asincrono.

Identificativo. Un identificatore è associato a ciascun aglet. Questo identificatore è univoco, globale e immutabile per tutta la durata dell'aglet.

Fondamentalmente ci sono solo due modi per dar vita ad un aglet: o è istanziato da zero (creation) o è copiato da un aglet esistente (clonazione). Per il controllo della popolazione si può ovviamente distruggere un aglet (disposal). Gli aglet sono mobili in due diversi modi: attivo e passivo. Nel primo un aglet migra di sua iniziativa dal suo host corrente a un host remoto (dispatching). Il modo passivo invece avviene quando un aglet viene spostato dall'host in cui risiede per mano di host remoto (retracting). Quando un aglet è in esecuzione richiede risorse, per ridurre il consumo di risorse, l'aglets può addormentarsi temporaneamente, rilasciando le risorse (deactivation), per poi tornare in esecuzione (activation) in un secondo momento. Infine, aglet multipli possono scambiarsi informazioni per compiere un determinato compito (messaging).

Questo è solo l'insieme minimo di operazioni necessarie per creare e gestire un ambiente distribuito di agenti mobili. Quando sono state create le Aglet API, uno degli obiettivi era quello di creare API leggere, facili e sufficientemente complete e robuste per le reali applicazioni. Qui di seguito riassumiamo le operazioni fondamentali degli aglets (vedi anche figura 1):

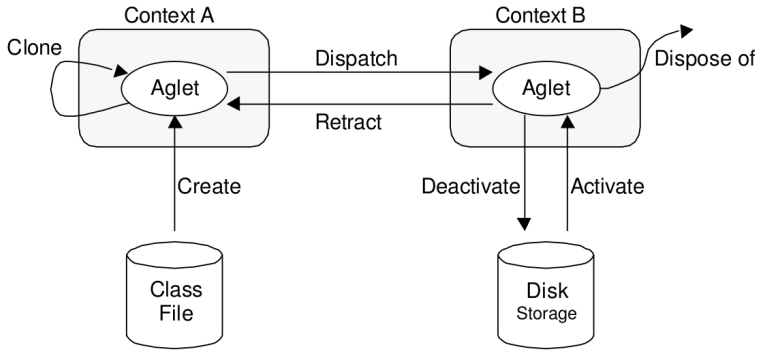

Figura 1: Modello del ciclo di vita di un aglet

Creation. La creazione di un aglet avviene in un contesto, al nuovo aglet viene assegnato un identificativo, poi viene inserito nel contesto, e inizializzato. L'aglet inizia l'esecuzione non appena è stato inizializzato correttamente.

Cloning. La clonazione di un aglet produce una copia quasi identica dell'aglet originale nello stesso contesto. Le uniche differenze sono l'identificativo assegnato e il fatto che l'esecuzione riprende nell'aglet nuovo. Si noti che i thread di esecuzione non vengono clonati.

Dispatching. Un aglet verrà rimosso dal suo attuale contesto e inserito nel contesto di destinazione, dove verrà riavviata l'esecuzione (i thread di esecuzione non migrano).

Retraction. Un aglet verrà rimosso dal suo contesto attuale e inserito nel contesto da cui è stata richiesta la retraction.

Attivazione e disattivazione. La disattivazione di un aglet è la capacità di fermare temporaneamente la sua esecuzione e conservare il suo stato in memoria secondaria. L'attivazione di un aglet lo ripristinerà nel suo contesto.

Disposal. Un aglet arresterà la sua esecuzione corrente e verrà rimosso dal suo contesto corrente.

Messaging. La messaggistica tra aglets comporta l'invio, la ricezione e la manipolazione di messaggi sincroni e asincroni.

2.2 JADE come framework per la costruzione di Mobile Agent

Java Agent DEvelopment Framework, o JADE, è un framework sviluppato in Java che supporta lo sviluppo di applicazioni distribuite e basate sul paradigma di programmazione ad agenti, fornendo un insieme di servizi di base, conformi allo standard FIPA e necessari alla creazione e al mantenimento di un sistema multiagente.

Ogni istanza dell'ambiente runtime di JADE è chiamata container ed all'interno dei container possono essere attivi uno o più agenti. Un insieme di container costituisce la platform. All'interno di ogni piattaforma deve sempre essere attivo uno speciale container, chiamato main container (container principale). Il main container è, inoltre, il primo container ad essere attivato alla partenza della piattaforma e tutti gli altri container si collegano ad esso al loro avvio. All'interno del main container, in conformità a quanto previsto da FIPA, sono presenti degli agenti con ruoli speciali deputati alla gestione della piattaforma stessa ed in particolare (vedi figura 2):

- Agent Management System (AMS): è il supervisore della piattaforma, responsabile delle operazioni di creazione e terminazione di agenti e container e dell'autenticazione e registrazione degli agenti con id univoco. Mantiene inoltre un elenco di tutti gli agenti che in un certo istante risiedono nella piattaforma.
- Directory Facilitator (DF): fornisce un servizio mediante cui un agente può pubblicizzare i propri servizi e/o ricercare servizi offerti da altri agenti.

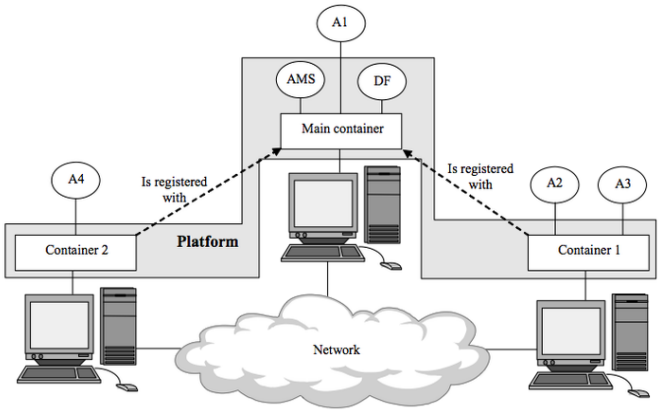

Figura 2: Architettura di JADE

JADE è stato sviluppato completamente in Java e, pertanto, la creazione di un agente in JADE corrisponde alla definizione di una classe che estende la classe

jade.core.Agent. Gli agenti attivi all'interno di un container JADE sono responsabili dell'esecuzione dei compiti loro assegnati. In JADE i compiti assegnati ad un agente sono modellati mediante un'astrazione chiamata behaviour (comportamento). Il programmatore può definire specifici behaviour ed assegnarli agli agenti di una piattaforma estendendo la classe jade.core.behaviours.Behaviour. Sono previste due tipologie di behaviour, realizzate come sottoclassi della classe behaviour: i SimpleBehaviour ed i CompositeBehaviour.

Gli agenti devono essere in grado di comunicare gli uni con gli altri, per cooperare, collaborare, negoziare e così via. Il modello di comunicazione adottato da Jade è l'Asynchronous Message Passing in cui ad ogni agente è associata una coda di messaggi ricevuti dagli altri agenti, aggiornata ogniqualvolta arrivi un nuovo messaggio. Le modalità e le tempistiche di recupero dei messaggi sono delegate alla logica applicativa dei singoli agenti. Il formato dei messaggi scambiati dagli agenti segue le specifiche del linguaggio ACL (Agent Communication Language) definito da FIPA.

Affinché gli agenti di una piattaforma possano comunicare, è necessario che essi condividano non solo linguaggi e protocolli di comunicazione, ma anche un comune vocabolario di termini utilizzati nei messaggi scambiati. La definizione di tale vocabolario in JADE è basata sul concetto di ontologia che trova frequenti applicazioni nell'ambito dell'intelligenza artificiale e della rappresentazione della conoscenza. Un'ontologia rappresenta un modo formale per definire un vocabolario comune di concetti relativi ad un preciso dominio, nonché le relazioni che intercorrono tra di essi.

JADE-Leap è una versione modificata di JADE che consente l'esecuzione di Agenti JADE su un'ampia gamma di dispositivi che varia dai server ai dispositivi mobili, questo è molto utile per lo sviluppo di Mobile Agent e in particolare per creare Mobile Agent che si occupano del Network Managment. JADE infatti è molto indicata per creare qualsiasi tipo di Mobile Agent, la piattaforma ha versioni compatibili che funzionano in diversi sistemi operativi, come Unix e Windows e le loro variazioni per i dispositivi portatili, il linguaggio di programmazione supportato è Java con tutti i suoi benefici di portabilità ecc., inoltre, l'integrazione con LEAP offre il supporto per i dispositivi portatili.

Il concetto di comportamento (behaviour) è molto utile, dato che può essere dinamicamente aggiunto ad un agente per cambiare le sue azioni e aggiungere nuove funzionalità. Inoltre, i comportamenti incapsulano le azioni degli agenti, questa caratteristica può essere utilizzata ad esempio per impostare o modificare l'itinerario di un Mobile Agent nella rete senza alterare l'agente corrispondente. Un altro vantaggio di JADE è la possibilità di verificare se una destinazione è a disposizione prima di passare un agente ad essa. Anche l'utilizzo di ACL per lo scambio di messaggi è molto utile, dato che non richiede di avere riferimenti concreti ad altri agenti per poter interagire con loro, l'interazione tra gli agenti può essere effettuata mediante lo scambio di messaggi asincroni.

Un inconveniente di JADE è quello di non fornire un supporto alla migrazione tra i diversi ambienti di esecuzione, pertanto, la migrazione tra i diversi host è supportata solo se i contenitori sono nella stessa piattaforma.

2.3 Perchè utilizzare i Mobile Agent nella gestione delle reti

I vantaggi riscontrati nell'utilizzo dei Mobile Agent possono essere così sintetizzati:

- Facilità nel modificare le funzioni di gestione esistenti. Ciò è dovuto al fatto che le funzioni di gestione sono sviluppate / modificate centralmente, e le modifiche hanno un'entrata in vigore immediata. Nel caso degli agenti statici invece, ciascuno di essi deve essere aggiornato da un messaggio di aggiornamento trasmesso dal gestore ai dispositivi gestiti, in questo caso frequenti modifiche creerebbero una notevole quantità di traffico.
- L'uso efficiente delle risorse di calcolo sulle entità gestite. Le funzioni di gestione vengono effettuate solo a condizione che il Mobile Agent risieda e sia attivo sul dispositivo da gestire.
- I Mobile agent sono in grado di fornire tutte le funzionalità offerte da parte dagli agenti statici, avendo l'ulteriore vantaggio della mobilità. In questo senso, un Moblie Agent può essere inviato al dispositivo gestito e resterà lì finché sarà necessario.
- La mobilità dei Mobile Agents implica di per sé una vista a livello globale della rete gestita. Questo permette ai Mobile Agents di applicare una seconda fase di gestione dei dati e di filtraggio a livello di dominio o di rete, cioè consente di confrontare/unire i risultati acquisiti dagli agenti su ogni host con quelli già raccolti durante il loro percorso e mantenere solo i valori conformi a talune restrizioni, portando ad una ulteriore riduzione dei trasferimenti di dati, in quanto solo una piccola parte dei dati originariamente acquisiti vengono inviati al gestore. Questo toglie una notevole mole di lavoro dall'host manager.
- La banda occupata per la migrazione dei Mobile agents non è necessariamente alta.

2.4 Schemi per il trasferimento del codice

Una questione chiave che influenza le prestazioni di una infrastruttura a Mobile Agent per il Network Managment è la dimensione dell'entità che viaggia tra gli host. In un tipico Mobile Agent framework basato su Java, sia il codice sia lo stato dell'agente sono richiesti dalla destinazione per creare un'istanza degli oggetti del Mobile Agent ricevuto. Sono stati identificati tre schemi alternativi per il trasferimento del codice:

- push: il codice viene inviato da una repository a tutti i dispositivi gestiti.
- pull: il codice viene caricato dal Network Element da una repository all'arrivo del Mobile Agent.
- migration: il codice viene inviato da un non-repository ad un altro.

Tutti i framework MA-based attuali utilizzano la migrazione come sistema di trasferimento del codice, vale a dire che il bytecode del Mobile Agent viene

trasferito insieme al suo stato in ogni trasferimento. Ciò comporta un maggiore carico sulle risorse di rete, in quanto la dimensione del codice è in genere molto più grande della dimensione dello Stato. Un'alternativa a questo può essere che il trasferimento del bytecode dell'agente venga eseguito una sola volta (a construction-time), mediante la trasmissione broadcast a tutti i dispositivi attivi gestiti, cioè usando lo schema push. Successivamente, è sufficiente il trasferimento del solo stato del Mobile Agent per far si che i dispositivi riconoscano l'arrivo dell'agente e lo attivino.

Schema Push per il trasferimento del codice. Il codice agente viene inviato alle stazioni presenti in un elenco. Di solito, il sistema di push è combinato all'utilizzo di itinerari predefiniti. Quando l'agente è avviato, il codice è inviato a tutte le stazioni del percorso. Così, quando l'agente arriva ad una stazione, il codice corrispondente è già lì e si può iniziare l'esecuzione immediatamente. Il regime di Push è realizzato mediante l'invio di messaggi, prima alla repository, che poi invia il codice alle stazioni (vedi figura 3).

Schema Pull per il trasferimento del codice. Lo schema pull è probabilmente il più diffuso, dal momento che viene utilizzato dai browser Web per scaricare le applet Java e script. Questo schema prevede che una stazione scarichi il codice agente corrispondente da una repository dopo che l'istanza agente è arrivata. Questo schema può essere applicato a Mobile Agent con percorsi flessibili in cui l'agente può decidere dinamicamente quale stazione sarà la successiva.

Lo svantaggio di questo sistema è un calo delle prestazioni. Quando un agente arriva ad una stazione, la sua esecuzione deve essere ritardata fino a quando il codice agente è stato scaricato completamente, il caching del codice dell'agente può evitare questo ritardo per arrivi successivi di agenti dello stesso tipo (vedi figura 4).

Schema Migrate per il trasferimento del codice. Questo schema è quello più ovvio per l'uso dei Mobile Agent. Il codice viene spostato insieme all'istanza dell'agente di stazione in stazione. Di solito, le performance sono inferiori rispetto allo schema Push, ma più alte rispetto allo schema Pull. Rispetto alla Push il tempo di migrazione è più lungo, perché nella Migrate devono essere trasferiti sia l'istanza (o lo stato) dell'agente sia il codice. Rispetto alla Pull la comunicazione con una repository è richiesta solo quando un agente viene lanciato. Dopo il lancio, l'agente è più flessibile perchè è indipendente dalla la disponibilità della repository (vedi figura 5).

Dato che ci sono vantaggi e lacune in tutti questi tre sistemi di distribuzione, quale sistema sia preferibile dipende dall'applicazione e dal tipo di attività che il Mobile Agent deve compiere. La disponibilità di più di uno schema offre inoltre una maggiore flessibilità per la gestione della cache; tutte le stazioni forniscono il caching del codice degli agenti al fine di migliorare le performance e per ridurre il carico di comunicazione sulla rete, ma dal momento che la dimensione della cache è limitata, il codice in cache deve essere cancellato di tanto in tanto. La gestione della cache delle stazioni e la scelta di quale schema utilizzare sono aspetti da tenere in seria considerazione per il corretto funzionamento del sistema.

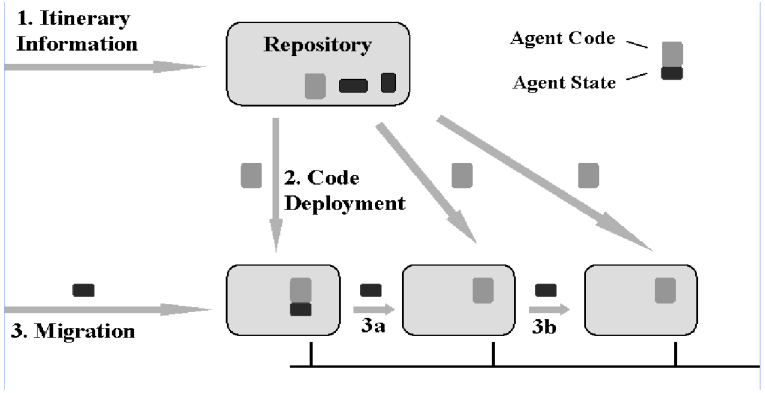

Figura 3: Distribuzione del codice mediante push

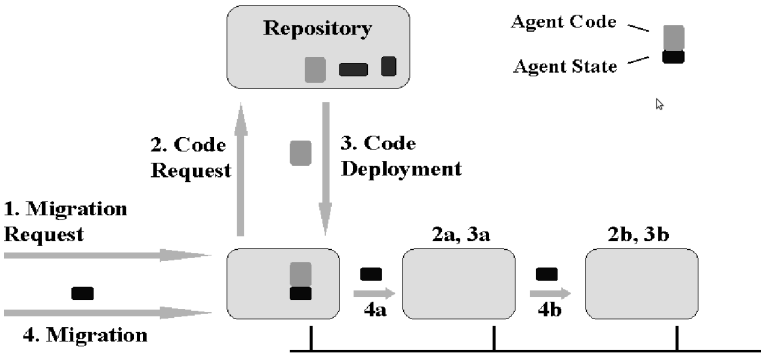

Figura 4: Distribuzione del codice mediante pull

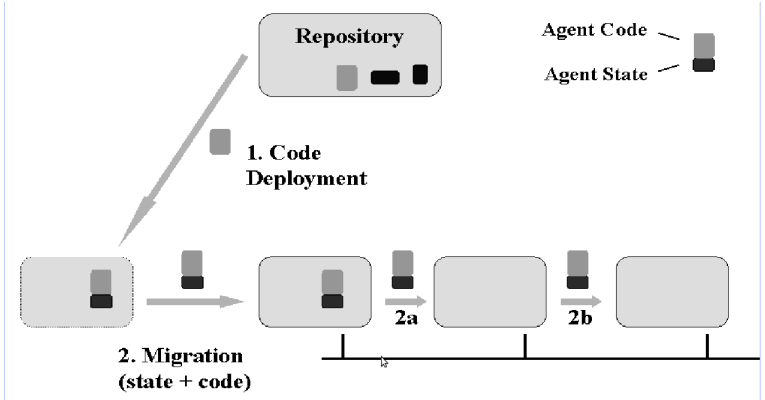

Figura 5: Distribuzione del codice mediante migrate

2.5 Mobile Agents Framework per il Network Managment

In questa sezione sarà illustrato un esempio di Mobile Agent Framework che usa push come schema di trasferimento del codice. Il framework è costituito da quattro elementi principali come illustrato nella figura 6.

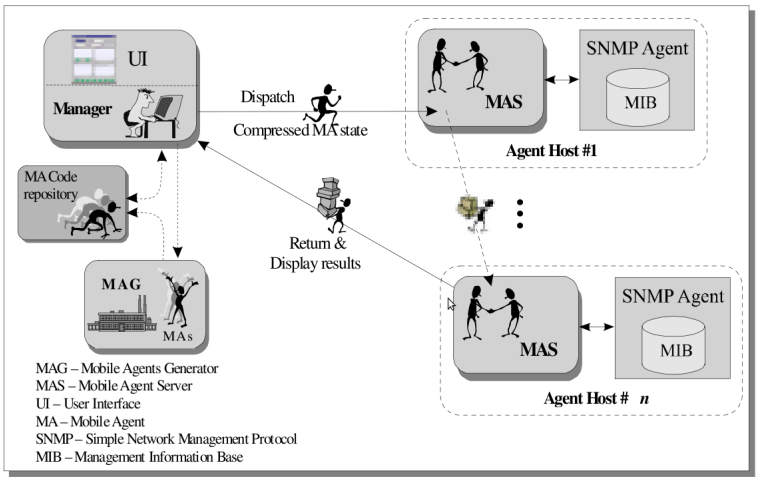

Figura 6: Infrastruttura del Framework

1. Manager application: è dotato di un'interfaccia Utente stile browser, coordina il monitoraggio e politiche di controllo riguardanti gli elementi della rete, mantiene e aggiorna la lista degli agenti attivi.

2. Mobile Agent Server: L'interfaccia tra i mobile agents e sistemi di gestione preesistenti è implementata nei moduli del Mobile Agent Server, installato su tutti i dispositivi gestiti. Il Mobile Agent Server risiede logicamente sopra l'agente SNMP standard, creando un efficiente ambiente run-time per la ricezione, l'istanza, l'esecuzione, e la spedizione degli oggetti dei mobile Agents. L'integrazione con SNMP è di vitale importanza per mantenere la conformità con i sistemi di gestione esistenti. La sua struttura può essere rappresentata come in figura 7.

3. Mobile Agent Generator: è essenzialmente uno strumento per la generazione automatica del codice dei Mobile Agents consentendo la costruzione di agenti personalizzati in risposta alle esigenze del particolare servizio. Il codice generato è memorizzato in un repository (vedi Figura 6). I mobile Agents possono dinamicamente estendere le loro funzionalità anche dopo l'inizializzazione del sistema, in modo da svolgere compiti di gestione adeguati alle esigenze di un ambiente in continua evoluzione come quello della rete.

4. Mobile Agents: sono oggetti Java con un ID univoco, in grado di migrare tra gli host in cui sono eseguiti come thread distinti per svolgere i loro specifici compiti di gestione. I Mobile Agent vengono forniti di una tabella di percorso, una cartella in cui vengono raccolti i dati di gestione e diversi metodi per controllare l'interazione con i dispositivi intervistati. Le informazioni di stato sono compresse prima di essere trasferite all'host di destinazione successivo.

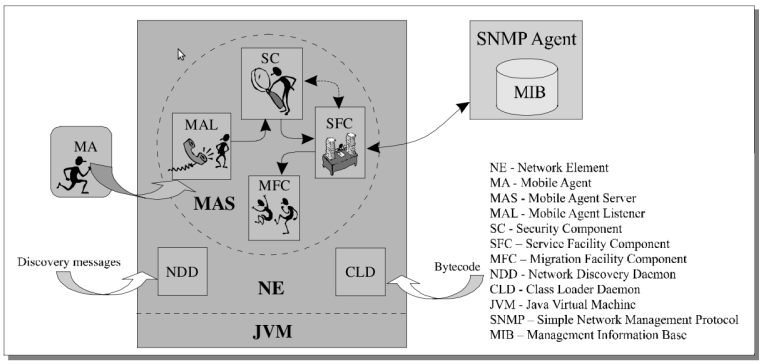

Figura 7: Struttura del Mobile Agent Server

2.6 Mobile agents e SNMP

Simple Network Management Protocol (SNMP) appartiene alla suite di protocolli Internet definita dalla IETF (Internet Engineering Task Force). Il protocollo opera al livello 7 del modello OSI e consente la gestione e la supervisione di apparati collegati in una rete. I tre componenti fondamentali del framework SNMP sono:

- sistema gestito (managed object)
- agente di gestione (management agent)
- sistema di gestione (manager)

Ogni sistema gestito (per esempio un semplice nodo, un router, una stampante o qualsiasi altro dispositivo che fornisca un'interfaccia di gestione SNMP) ospita un agente di gestione (master agent) e solitamente un certo numero di subagent. Il master agent ha il ruolo di intermediario fra il manager (che è l'applicazione remota che prende le decisioni di gestione, per esempio sotto il controllo diretto dell'operatore umano) e i subagent (che sono gli esecutori di tali decisioni). Ciascun subagent è incaricato di attuare le decisioni di gestione nel contesto di un particolare sottosistema o relativamente a un particolare aspetto del sistema gestito.

SNMP utilizza una chiara separazione fra il protocollo di gestione e la struttura dell'oggetto gestito. Nell'architettura SNMP, per ogni sottosistema è definita una base dati detta MIB (Management Information Base), gestita dal corrispondente subagent, la quale rappresenta lo stato del sottosistema gestito, o meglio, una proiezione di tale stato limitata agli aspetti di cui si vuole consentire la gestione. Ogni modifica alla MIB causa un corrispondente mutamento nello stato del sottosistema rappresentato, e viceversa. Garantire questa proprietà della MIB è la funzione principale del subagent che la gestisce.

L'accesso alla MIB (in lettura e scrittura) rappresenta l'interfaccia fornita al manager per gestire il sistema. Ogni MIB, pur variando nei contenuti specifici, ha la medesima struttura generale e i medesimi meccanismi generali di accesso da parte del manager (lettura e scrittura dei dati), è quindi possibile al manager

agire sullo stato del sottosistema in maniera indipendente dalle procedure concrete che devono essere messe in atto (dal subagent) per estrarre le informazioni di stato rappresentate nella MIB, o attuare le modifiche di stato a seguito di cambiamenti dei contenuti della MIB, prescindendo dai dettagli di come una tale modifica venga poi concretamente attuata sul sistema gestito. Più in dettaglio, il manager dialoga con i sistemi gestiti essenzialmente in due modi: invia richieste SNMP e riceve notifiche SNMP.

Questo protocollo utilizza perciò agenti pressochè stazionari e esplica la gestione delle risorse di rete attraverto la comunicazione e lo scambio di messaggi tra gli agenti. I Mobile Agent forniscono un ulteriore passo in avanti rispetto agli agenti stazionari, è percio possibile sfruttare le caratteristiche dei Mobile Agent per migliorare le prestazioni del SNMP.

E' stato preso in considerazione proprio questo protocollo perchè SNMP è probabile che sia il protocollo di gestione di rete scelto per il futuro, e molti nodi della rete saranno dotati di un agente SNMP. Alcuni di questi agenti possono essere agenti preesistenti anziani che devono essere rafforzati con una più moderna strumentazione aggiuntiva. Anche se i Mobile Agent verranno impiegati per le nuove attività, può essere ragionevole sfruttare anche le capacità degli agenti SNMP già esistenti. In base a questo concetto, consideriamo il tipo di interazione più probabile tra agenti SNMP e Mobile Agent:

Un Moblie Agent potrebbe arrivare a un nodo e volere accedere ai dati dal MIB SNMP residente. Si potrebbe dotare il Mobile Agent di capacità di gestione SNMP e consentirgli di rilasciare i pacchetti di richiesta SNMP per l'agente SNMP presente, come se fosse un manager. Per fare ciò occorre dotare l'agente SNMP di un'interfaccia sicura creata apposta per questo scopo. Il vantaggio di tale approccio è che non si deve modificare l'Agente SNMP per fornire l'accesso ai propri MIB, ma il costo di questo è che il Mobile Agent ha bisogno di avere la piena capacità di gestione SNMP.

Un Moblie Agent potrebbe arrivare a un nodo e voler estendere la capacità dei SNMP MIB esistenti, questo perchè i Mobile Agent sono in grado di portare con se (con il loro stato) alcune informazioni provenienti dall'esterno e dalla rete in generale che è stata analizzata fino a quel punto. Il protocollo SNMP DPI e il suo successore il protocollo AgentX sono stati progettati appositamente per consentire ad un agente di estendere o migliorare un agente SNMP esistente.

3 INCA

In questa sessione viene descritta l'architettura di INCA, che è progettata per la gestione distribuita di reti multi-servizio e dei sistemi di applicazioni. L'Intelligent Network Control Architecture è popolato da agenti intelligenti stazionari e mobili. Questi agenti effettuano il monitoraggio e il controllo della rete e dei componenti, sostenendo in tal modo la gestione integrata sia delle reti che dei servizi. L'architettura prevede funzionalità di transazione per il controllo dei trasporti e della mobilità degli agenti, gestione delle priorità, e molteplici sistemi di trasferimento del codice degli agenti. Gli oggetti utilizzati per accedere alle risorse di elementi di rete e nuove funzionalità di sistema possono essere creati, distribuiti, e sostituiti dinamicamente. Il design di questa piattaforma ad agenti per la gestione di rete, non è vincolato a un particolare linguaggio

di programmazione o di ambiente di elaborazione, la corrente implementazione, tuttavia, si basa su Java e RMI.

Qui vengono impiegati Mobile Agents a sostegno della mobilità dell'operatore di rete, però, il sistema centrale di gestione della rete si basa su una gerarchia degli agenti stazionari.

Come già spiegato la gestione della rete oggi è generalmente basata su un modello client-server con SNMP e CMIP come standard de facto per il monitoraggio e per la gestione di una rete di computer e dispositivi. Gli oggetti gestiti, come host, router, switch, bridges, stampanti, ecc, forniscono l'accesso in lettura/scrittura di una serie di variabili attraverso un processo di gestione. Una stazione di gestione può quindi comunicare con tali processi mediante un paradigma client-server. Anche se semplice, questo approccio centralizzato per la gestione della rete ha alcuni gravi svantaggi per quanto riguarda la scalabilità, flessibilità e prestazioni. Dividendo le funzioni di gestione in enti di mobile computing, autonomi e intelligenti (vale a dire agenti software), molti dei problemi della rete e della gestione del servizio possono essere superati. La scalabilità aumenta, la gestione non viene eseguita esclusivamente dalla stazione di gestione, ma delegata ai vari agenti di gestione distribuita. I compiti ripetitivi possono essere evitati se gli agenti software sono in grado di imparare dall'esperienza. La tolleranza all'errore può essere incrementata attraverso l'autonomia e l'apprendimento di nuove capacità da parte degli agenti di gestione. Performance migliori si ottengono spostando la funzionalità di gestione più vicino all'effettivo elemento di rete, riducendo così il traffico di rete. I dettagli di basso livello dei dispositivi diversi possono essere nascosti dietro l'interfaccia agente.

L'implementazione corrente di INCA è basata sul linguaggio Java. Java supporta già la mobilità del codice e il caricamento dinamico delle classi, tuttavia, il Java class loader è stato sostituito in modo da essere in grado di caricare il codice dalla stazione repository. Come infrastruttura di comunicazione sono supportati Java RMI e CORBA.

INCA è una infrastruttura per supportare applicazioni per la gestione integrata di servizi di rete e nei settori delle telecomunicazioni. INCA in particolare è stato sviluppato per le applicazioni di gestione distribuite basate su Mobile Agent intelligenti.

Un'istanza della piattaforma INCA è costituito da un insieme di stazioni e da servizi offerti a livello locale, da una stazione, o a livello globale, dalla piattaforma.

3.1 Servizi locali

Una stazione è locale a un elemento di rete e fornisce i seguenti servizi:

Esecuzione concorrente di agenti. Agenti multipli - ognuno con un proprio thread di controllo - possono essere eseguiti contemporaneamente. Gli agenti sono schedulati in base alla loro priorità da uno schema di scheduling specifico di INCA. Nessun agente di minore priorità verrà eseguito se vi è un agente con priorità più alta in esecuzione.

Caricamento del codice dell'agente. Il bytecode degli Agenti può essere trasferito tra le stazioni. Tutti e tre i sistemi per il trasferimento del codice sono supportati: push, pull e migrate. Tutte le stazioni supportano lo stesso formato di codice, vale a dire che lo stesso codice può essere eseguito in qualsiasi host. Al fine di migliorare le prestazioni è utilizzata una cache del codice.

Trasferimento dello stato dell'agente. Lo stato attuale di un agente può essere trasferiti da una stazione all'altra, in modo da permettere la migrazione dell'agente.

Capacità di transazione. Per la comunicazione peer-to-peer tra le stazioni è offerto un layer affidabile e fault-tolerant chiamato transaction capabilities (vedi sezione 3.4).

Accesso alle risorse locali. L'accesso alle risorse locali in una stazione è fornita dai managed object. Questi managed object rappresentano gli elementi di rete (o parti di questi) da gestire, la loro interfaccia specifica a quali risorse può accedere e come. Di solito, una stazione contiene almeno una istanza di un managed object che dà accesso alle risorse locali degli elementi di rete. Ulteriori managed object possono essere creati dinamicamente su richiesta di un agente.

Supporto per la localizzazione e monitoraggio dello stato degli agenti. La localizzazione e lo status di un agente che migra da una stazione all'altra può essere controllato da un 'istanza centrale, il Locator.

3.2 Global Services

Oltre ai servizi forniti da ogni singola stazione ci sono servizi offerti da tutta la piattaforma, o da un insieme di stazioni dedicate:

Controllo. Per ogni piattaforma INCA c'è almeno una stazione di controllo. Questa stazione esegue un agente stazionario di controllo che offre un servizio di controllo per l'avvio e il monitoraggio degli agenti. Di solito, la posizione di questa stazione è la console del gestore di rete.

Repository. Stazioni dedicate che offrono un servizio di repository. Questo servizio fornisce accesso al codice di tutti gli agenti che possono funzionare sulla piattaforma. La repository è utilizzata in base allo schema che viene usato (pull, push o migrate).

Locator. Su una stazione unica gira un agente stazionario (il Locator) che offre il servizio della tracciabilità, l'agente localizzatore riceve messaggi dalle stazioni sugli agenti monitorati. Sulla base di queste informazioni l'agente locator risponde alle richieste dell'agente di controllo sulla posizione e lo stato degli altri agenti. Inoltre supervisiona gli itinerari predefiniti.

Itinerario di controllo. Il servizio di controllo consente di attribuire ad un agente mobile un itinerario predefinito. L'itinerario entra a far parte dello stato dell'agente, ma non è accessibile, è controllato da una stazione ogni volta che un Mobile Agent si sposta per determinare la stazione successiva. Quando un agente con un percorso predefinito viene lanciato dall'agente di controllo, una copia del percorso viene inviata al locator che mette a confronto l'itinerario con i messaggi location.

Controllo delle migrazioni. Combinando il controllo dell'itinerario con la persistenza dello stato dell'agente la fault-tolerance nella migrazione degli agenti è molto aumentata (vedi sezione 3.5).

Comunicazione inter-agent. La comunicazione tra gli agenti avviene attraverso un'interfaccia comune, questa funzionalità include un naming service presso il quale gli agenti possono ottenere riferimenti ad altri agenti.

3.3 Priorità

L'esecuzione contemporanea di più agenti non solo aumenta le funzionalità di una piattaforma ma introduce anche il problema dello scheduling degli agenti. Si è osservato che uno scheduling equo di tutti gli agenti del sistema non è l'ideale per le applicazioni di gestione della rete. Di solito, gli agenti di controllo di rete devono essere eseguiti velocemente, perché il gestore della rete è in attesa della loro esecuzione, al contrario, gli agenti di monitoring, in genere sono attivi per un lungo tempo e devono essere interrotti quando uno degli agenti dell'altro tipo deve essere eseguito. Per soddisfare tali requisiti sono state usate le priorità. Un agente di priorità più alta interrompe un agente di priorità più bassa e nessun agente di priorità inferiore viene eseguito mentre un agente di priorità più alta è in esecuzione e non è bloccato.

3.4 Comunicazione tra gli Agenti

In INCA la comunicazione tra gli agenti può essere rivista come comunicazione peer-to-peer tra le stazioni, in quanto i Mobile Agents risiedono su host differenti. Questo servizio è implementato sul layer chiamato Transaction Capabilities che risiede tra i livelli di middleware e di application. Per permettere la comunicazione questo layer fornisce l'importante servizio del naming e permette una comunicazione trasparente tra gli oggetti distribuiti, cioè in un ottica più ampia permette lo scambio di messaggi tra 2 peer comunicanti. Questa è la tecnica usata da INCA per ottenere uno scambio di messaggi affidabile tra agenti distribuiti.

3.5 Controllo della migrazione

Quando si avvia un agente con un itinerario predefinito, ne viene archiviata una copia presso il locator. Un fallimento presso la stazione che ospita il Mobile Agent non porta necessariamente ad una rottura del compito dell'agente, in quanto una copia aggiornata del percorso dell'agente viene mantenuta memorizzata nel locator, perciò può essere creato un nuovo agente che visita le stazioni restanti del percorso. In contrasto con questo concetto, i Mobile Agents potrebbero non fare uso di un itinerario predefinito, ma determinare gli host da visitare in modo dinamico, a run-time. In ogni caso, i Mobile Agents che richiedono particolare robustezza è meglio che facciano uso di itinerari predefiniti.

Se un Mobile Agent non fosse dotato di un itinerario predefinito, di fronte a un guasto di una stazione intermedia tutto il suo lavoro andrebbe perso e la piattaforma sarebbe costretta a creare una nuova istanza dell'agente e a ricominciare da capo, ma utilizzando il locator e l'itinerario è possibile, di fronte ad un guasto, ripristinare l'agente e farlo ripartire da dove era arrivato il suo predecessore prima del fallimento, così si evitano sovrascritture e un eccessivo carico del sistema. Ma se un agente è più complicato, ed è dotato anche di uno stato, la conoscenza del solo punto di guasto nell'itinerario non è sufficinte a ripristinare il lavoro interrotto. Perciò lo stato dell'agente deve essere conservato da ogni stazione prima della migrazione dell'agente, in questo modo, quando avviene un guasto su una stazione, il time-out del locator scade senza che egli abbia ricevuto il messaggio di arrivo del Mobile Agent, il locator così capisce che cè stato un errore e creerà una nuova istanza dell'agente che potrà

ricominciare da dove era avvenuto il fallimento e con lo stato memorizzato nella cache dell'ultimo host visitato.

4 JAMES

JAMES è una piattaforma Java-Based per Mobile Agent, l'obiettivo principale di questa piattaforma è la gestione della rete e delle applicazioni di telecomunicazione integrando al suo interno anche il supporto per SNMP. Il progetto JAMES esplora l'uso della tecnologia dei Mobile Agent nelle applicazioni di Network Managment e si compone di due linee complementari di lavoro. In primo luogo, lo sviluppo di una piattaforma per Mobile Agent progettata specificatamente in base ai requisiti che applicazioni impongono. In secondo luogo, lo sviluppo dei prodotti software che sfruttano i vantaggi tecnologici ed economici della piattaforma.

La piattaforma JAMES fornisce l'ambiente per l'esecuzione di Mobile Agents. Nella sua attuale versione, vi è una distinzione tra l'ambiente software che viene eseguito nell'host Manager (James Manager) e il software che viene eseguito in negli altri nodi della rete (designato come JAMES Agency). La Figura 8 mostra una visione globale del sistema.

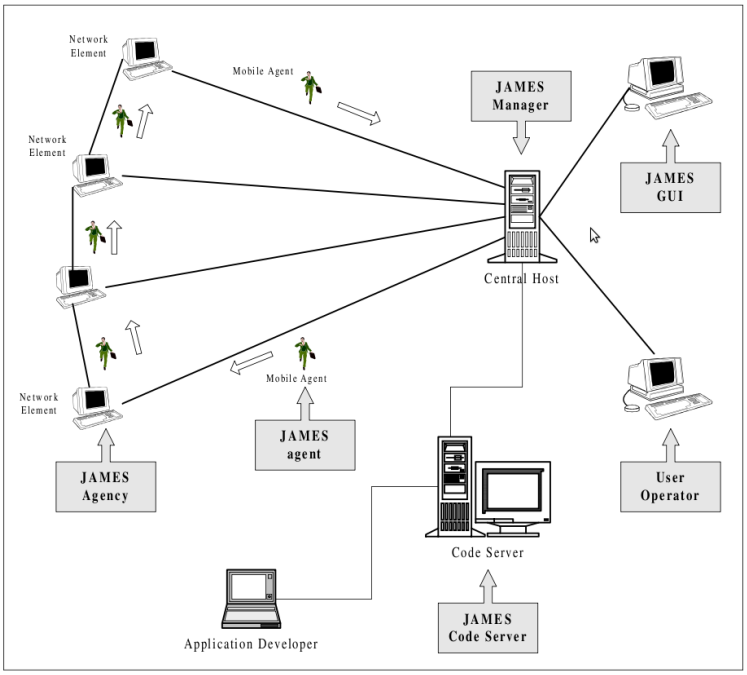

Figura 8: Architettura di James

Il James Manager è responsabile della manutenzione di tutta l'infrastruttura, ogni elemento della rete gestisce una Java Virtual Machine ed esegue un JAMES Agency che permette l'esecuzione dei Mobile Agent. Gli agenti migrano attraverso queste macchine della rete per accedere a dati, eseguire alcuni com-

piti e per scambiare informazioni con altri agenti o altre applicazioni. C'è un meccanismo di autenticazione nelle JAMES Agency per controllare l'esecuzione degli agenti e per evitare l'intrusione di agenti non autorizzati. Le applicazioni sono costituite da uno o più Mobile Agent, e possono includere anche classici programmi statici per coordinare le attività dei Mobile Agent o fornire un Interfaccia Grafica per l'utente finale. Questi moduli statici esterni comunicano con gli agenti utilizzando API remote, che consentono anche la gestione remota della piattaforma. I Mobile Agent sono sviluppati in Java e utilizzano delle API specifiche di JAMES per il controllo della mobilità.

In James un Mobile Agent è in grado di migrare verso alcuni computer remoti, dove esegue una funzione o raccoglie alcuni dati rilevanti e poi migra su altre macchine, al fine di svolgere un altro compito.

C'è una distinzione tra l'ambiente software che viene eseguito in host manager e il software che viene eseguito in elementi di rete: l'host centrale esegue il James manager, mentre i nodi della rete di eseguono una JAMES agency. Il sistema JAMES fornirà un'interfaccia di programmazione che consente la manipolazione e la creazione da parte dei programmatori di Mobile Agent (figura 8).

Un protocollo speciale è stato sviluppato per trasferire gli agenti attraverso le macchine in modo robusto e per garantre atomicità al verificarsi di guasti. Lo sviluppatore delle applicazioni le scrive basandosi su una serie di Mobile Agent, queste applicazioni sono scritte in Java e usano le API JAMES per il controllo della mobilità. La macchina host che esegue il James Manager è responsabile della gestione dell'intero sistema di Mobile Agent. Esso fornisce l'interfaccia per l'utente finale per il controllo e il monitoraggio remoto degli agenti e delle applicazioni.

Ogni agente Mobile ha uno specifico itinerario con una serie di compiti (Missioni) che sono eseguiti attraverso le JAMES Agency. L'agente è solitamente creato e spedito dal James manager. Le classi del codice degli agenti JAMES sono raggruppate in file JAR, uno per ogni agente. Il primo passo è quello di registrare gli agenti nella piattaforma, tutti i JAR che sono stati registrati sono memorizzati nel Code Store. Le JAMES Agency hanno un proprio disco locale usato come cache per il codice degli agenti che si avvale della politica LRU per la sostituzione dei jar quando la cache è piena. Ogni volta che un agente arriva a una agenzia le sue classi sono ricercate nella cache in memoria, poi, nella cache del disco locale, se non sono presenti allora l'agente le caricherà dal JAMES agency precedente, se anche qui le classi non sono presenti allora verranno caricate dal Code Server.

Questo schema gerarchico di ricerca fornisce una grande flessibilità nella distribuzione del codice, c'è sempre un punto centrale dove tutto il codice rimane memorizzato.

Il caching dei file JAR cerca di sfruttare la località del codice di Mobile Agent. Tuttavia, si possono migliorare ulteriormente le prestazioni del sistema ottimizzando le parti interne del protocollo di migrazione attuando una tecnica di prefetching del codice per accelerare l'esecuzione degli agenti. Quando un agente viene creato e lanciato nella piattaforma JAMES, esso dispone di un proprio itinerario di JAMES agency. C'è sempre un overhead nel tempo di avvio in cui le classi dell'agente non si trovano nella cache locale delle JAMES agency. Questo tempo di avvio corrisponde al tempo che occorre per prendere le classi dal disco locale o da un'altra agenzia o dal Code Store, si può ridurre

questo tempo informando tutte le agenzie che alcuni particolari agenti stanno per essere eseguiti lì (Code Prefetching).

Periodicamente vengono salvate tutte le informazioni necessarie per rendere il sistema resistente a possibili guasti degli elementi di rete, il sistema deve avere abbastanza informazioni per recuperare uno stato precedente coerente. Questo si ottiene anche mediante un meccanismo di checkpoint, quando un Mobile Agent termina un compito in una JAMES Agency, il suo stato interno viene salvato prima che sia trasmesso alla destinazione successiva. L'agente è migrato verso un altro host, ma i suoi dati rimangono salvat fino a quando l'agente non è stato avviato con successo nella prossima Agency. Questo meccanismo checkpointing è trasparente agli sviluppatori di applicazioni ed è incorporato nel protocollo di migrazione per garantire l'atomicità del trasferimento degli agenti. Ciò significa che quando si verifica un fallimento l'agente non va perduto e il sistema è in grado di recuperare il suo stato nell'Agency precedente. Quando c'è un errore nel protocollo di migrazione o se una delle Agency nell'itinerario non è disponibile l'agente può eseguire uno delle seguenti tre procedure:

1. Tornare alla Gestione JAMES;
2. Andare all'Agency successiva disponibile nel percorso;
3. o semplicemente attendere che l'Agenzia di destinazione sia tornata attiva.

4.1 JAMES e SNMP

La piattaforma JAMES fornisce il supporto per SNMP in modo bi-direzionale (vedi figura 9), offrendo tre servizi distinti: l'interazione con gli Agenti SNMP locali e remoti, l'accesso a servizi basati su Mobile Agent da parte di applicazioni preesistenti, e un servizio che permette ad applicazioni già presenti di gestire la piattaforma stessa di JAMES.

Inoltre sono stati definiti alcuni obiettivi fondamentali di progettazione:

- Trasparenza: per ovvie ragioni, le applicazioni che già utilizzano correttamente SNMP non dovrebbero subire alcuna modifica;
- Impatto minimo sulle infrastrutture JAMES: l'accesso alle risorse di rete deve essere il più possibile indipendente dal fatto che esse supportino SNMP
- Pieno supporto per la mobilità degli agenti: il che significa che il supporto SNMP non dovrebbe limitare la migrazione degli agenti.

La progettazione dei servizi SNMP di James cerca di realizzare la definizione di una struttura modulare in cui ogni Servizio SNMP può essere installato e rimosso on-the-fly, secondo le esigenze. La maggior parte dei servizi consistono in Mobile Agent (i Service Agent) che forniscono servizi ai comuni Mobile Agent attraverso la comunicazione inter-agent. Questi Service Agent, che possono essere situati (o implicitamente installati) usando un generale servizio di directory, sono una soluzione leggera per la fornitura di diversi tipi di servizi.

L'SNMP-agent di JAMES permette la comunicazione SNMP tra le applicazioni esistenti che usano uel protocollo e gli agenti mobili, aprendo la strada a una facile installazione di nuovi servizi di gestione (corrispondenti a uno o più Mobile Agent) utilizzabili dai sistemi esistenti di gestione della rete. In un tale

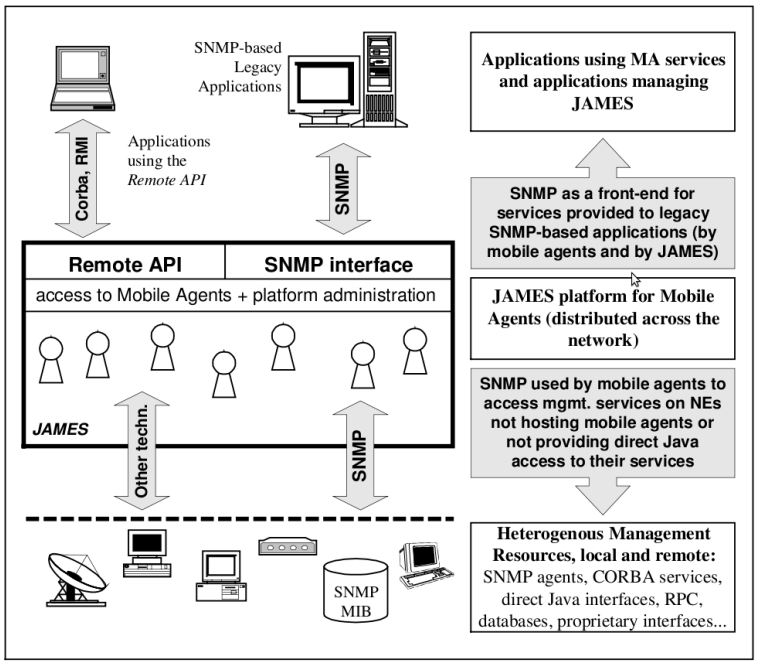

Figura 9: Approccio bidirezionale di JAMES a SNMP

scenario i Mobile Agent possono essere utilizzati per pre-elaborare i dati raccolti dai servizi di gestione esistenti (offrendo funzionalità di livello superiore), e possono anche installare, in modo dinamico, nuovi servizi di gestione.

Questo tipo di sostegno è importante ogni volta che SNMP è l'unica o la migliore interfaccia disponibile per l'accesso alla gestione delle informazioni. Un'altra ragione per fornire il supporto SNMP è la possibilità di utilizzare la tecnologia dei Mobile Agents per sviluppare nuovi servizi di gestione utilizzati da applicazioni di gestione esisitenti.

L'approccio JAMES prevede anche maggiore sostegno alla mobilità degli agenti e, attraverso l'utilizzo del concetto di Service Agent, presenta un impatto minimo sulla complessità e sulle prestazioni della piattaforma.

Riferimenti bibliografici

[1] www.wikipedia.org

[2] Advanced Network Monitoring Applications Based on Mobile/Intelligent Agent Technology *Damianos Gavalas, Dominic Greenwood, Mohammed Ghanbari, Mike O'Mahony*

[3] Integration of Mobile agents with SNMP: Why and How *B. Pagurek, Y. Wang, and T. White*

[4] Mobile Software Agents for Network Monitoring and Performance Management *Damianos Gavalas*

[5] INCA: An Agent-based Network Control Architecture *J. Nicklisch, J. Quittek, A. Kind, S. Arao*

[6] ENABLING MOBILE AGENT TECHNOLOGY FOR LEGACY NETWORK MANAGEMENT FRAMEWORKS *Paulo Simões, Rodrigo Reis, Luís M. Silva, Fernando Boavida*

[7] JAMES: A Platform of Mobile Agents for the Management of Telecommunication Networks *Luis Moura Silva, Paulo Simoões , Guilherme Soares, Paulo Martins, Victor Batista, Carlos Renato, Leonor Almeida Norbert Stohr*

[8] Mobile Agents with Java: The Aglet API *Danny B. Lange, Mitsuru Oshima*

[9] Implementing Mobile Agent Design Patterns in the JADE framework *Emerson Ferreira de Araújo Lima, Patrícia Duarte de Lima Machado, Jorge César Abrantes de Figueiredo, Flávio Ronison Sampaio*