INDICE

INTRODUZIONE

La crescente proliferazione di dispositivi portatili che possono fruire di connettività wireless, i recenti sviluppi delle tecnologie wireless e l'emergenza delle Mobile Ad-hoc NETwork (MANET), aprono un nuovo scenario. Nel nuovo scenario gli utenti non richiedono solo la possibilità di fruire dei tradizionali servizi Internet, quali il web o la e-mail, ma richiedono anche la possibilità di beneficiare di servizi collaborativi avanzati. I nuovi servizi devono consentire agli utenti di collaborare in modo impromptu ovunque essi si trovino, in qualunque momento e quando sono in movimento. Esempi di servizi abilitati dalle tecnologie MANET sono costituiti da applicazioni per il file sharing fra utenti mobili, dal coordinamento di veicoli, da scenari di protezione civile.
Le caratteristiche delle MANET sollevano molti problemi nello sviluppo di servizi collaborativi. La topologia della rete non è determinabile a priori rendendo impossibili assunzioni sulla corrente disponibilità on-line delle diverse entità interagenti. Disconnessioni, partizioni e merge di rete sono eventi comuni che causano transitori nella collaborazione fra partner nuovi e precedentemente sconosciuti. Inoltre, un ulteriore elemento di complessità deriva dalla richiesta degli utenti di potere collaborare tramite dispositivi eterogenei sia per natura che per prestazioni quali lap-top, PDA o telefoni cellulari.
L'alto livello di dinamicità degli scenari MANET mina le assunzioni di progetto alla base delle soluzioni di supporto per servizi collaborativi attualmente disponibili. Infatti, questi sistemi tipicamente richiedono la disponibilità di terminali con elevate risorse computazionali, la disponibilità di canali di comunicazione affidabili dotati di larga banda e non si adattano ai frequenti cambiamenti di posizione e di terminale dell'utente. Le tecnologie MANET richiedono perciò di ripensare e di riprogettare il supporto a servizi di tipo collaborativo.
In questa tesi verrà presentato AGAPE, un framework context-aware per la gestione dei gruppi che promuove e supporta lo sviluppo di servizi collaborativi in scenari MANET. AGAPE consente la creazione di gruppi on-demand, il discovery ed il monitoring della diponibilità on-line dei diversi gruppi e dei loro membri. Una importante peculiarità di AGAPE è quella di fornire a livello applicativo ad ogni entità la completa visibilità di tutti i membri del gruppo a cui appartiene, insieme ai loro attributi e caratteristiche.

La tesi è organizzata come segue: il *Capitolo 1* descrive le principali caratteristiche delle reti wireless e fornisce dettagli sulle tecnologie di rete e sui protocolli di routing. Nel *Capitolo 2* viene presentato il modello di AGAPE. Il *Capitolo 3* mostra in dettaglio il componente VMS. Il *Capitolo 4* mostra un esempio applicativo che sfrutta i servizi di AGAPE. Seguono le conclusioni.

CAPITOLO 1

RETI WIRELESS

1.1 **Introduzione**

I recenti progressi nelle reti wireless ed il crescente successo dei dispositivi mobili, come laptop computer, Personal Digital Assistant (PDA), telefoni cellulari e tablet PC aprono un nuovo scenario in cui gli utenti richiedono di beneficiare di servizi internet in ogni momento ed indipendentemente dalla propria posizione fisica. Le tecnologie wireless risultano particolarmente adatte qualora sia necessario supportare la mobilità dei dispositivi utenti o per il deployment di infrastrutture di comunicazione in ambienti difficili.
Uno dei vantaggi principali di questa tecnologia riguarda infatti la facilità d'installazione che rende le reti wireless ideali per quegli ambienti dove non si intende intervenire sulle strutture murarie dell'edificio per installare cavi. Sono quindi ideali anche per installazioni "temporanee" (mostre, fiere, congressi, situazioni di emergenza, gruppi di lavoro).
L'assenza di cavi permette agli utenti di spostarsi rimanendo connessi alla rete; le reti wireless consentono così di abbattere alcuni limiti che caratterizzavano i tradizionali metodi di connessione, rendendo possibile la comunicazione di dispositivi in movimento (mobile computing).
Parlando di mobile computing bisogna considerare la rete come un insieme dinamico di nodi, liberi di muoversi all'interno di essa; un sistema in continua evoluzione in cui è necessario gestire i cambiamenti in modo appropriato e tempestivo.
Questo nuovo concetto di rete apre le porte a nuove problematiche e crea l'esigenza di rinnovamento anche a livello di infrastrutture software e middleware. La dinamicità degli ambienti wireless rende impossibile assunzioni a priori rispetto la topologia della rete stessa che può cambiare continuamente. Il problema più rilevante che ne deriva riguarda l'instradamento dei dati tra nodi; altre problematiche per questo tipo di rete sono la sicurezza della trasmissione via etere dei dati, il controllo degli accessi, l'assegnazione degli indirizzi IP ai nodi che si aggiungono alla rete, la possibilità di perdere link di connessione a causa del movimento dei nodi e la limitata larghezza di banda.

1.2 **Topologia delle reti wireless**

Come mostrato in figura 1.1, possiamo indicativamente classificare le reti wireless in base all'area coperta dal segnale trasmesso dai dispositivi, in diverse categorie: **BAN**, **PAN**, **WLAN** e **WWAN**.
Questa classificazione è largamente diffusa da tempo.

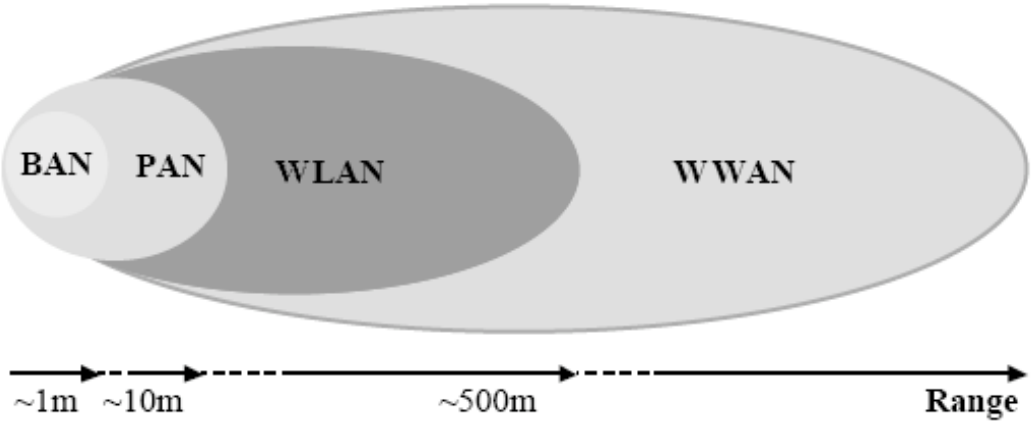


Figura 1.1 Topologia delle reti ad hoc.

1.2.1 **Body Area Network**

Le BAN sono reti che hanno un raggio di trasmissione che copre all'incirca le dimensioni di un corpo umano, tipicamente 1-2 metri, e consentono di connettere dispositivi indossabili quali auricolari, palmari, lettori MP3, telefoni cellulari etc.
Tra le caratteristiche principali delle BAN troviamo la capacità di connettere dispositivi eterogenei e la capacità di auto-configurarsi, rendendo trasparenti all'utente operazioni come la rimozione o l'aggiunta di un nuovo dispositivo da una BAN.
La connessione wireless è considerata la soluzione naturale per le BAN in quanto l'utilizzo di fili risulterebbe scomoda.
Uno dei primi esempi di BAN è il prototipo sviluppato da T.G.Zimmerman [1], che poteva fornire una comunicazione dati (con una data rate che arrivava fino a 400 Kbps) sfruttando il corpo come un canale. In particolare, Zimmerman mostrò che i dati possono essere trasferiti attraverso la pelle sfruttando una corrente molto bassa. Il trasferimento dati

tra due persone (ovvero l'interconnessione tra due BAN) poteva essere realizzata attraverso una semplice stretta di mano.

1.2.2 **Personal Area Network**

Il raggio di comunicazione delle PAN è tipicamente superiore ai 10 metri.
Le PAN consentono a dispositivi vicini di condividere dinamicamente informazioni. Mentre una BAN si dedica all'interconnessione dei dispositivi indossabili da una persona, una PAN è sviluppata nella immediata prossimità degli utenti.
E' possibile perciò connettere dispositivi portatili con altri oppure con stazioni fisse, ad esempio per accedere ad Internet.
Tecnologie ad infrarossi e radio rendono nelle PAN notevolmente più pratiche le operazioni quotidiane di sincronizzazione fra portatile, desktop e palmare, ma anche il download delle immagini dalla macchina fotografica digitale, l'upload di musiche nel riproduttore di MP3, ecc.
Tra gli standard più usati per realizzare le BAN e le PAN ci sono l'IrDA e il Bluetooth.
Il protocollo IrDA (Infrared Data Association) consente collegamenti bidirezionali point-to-point sfruttando gli infrarossi. L'interconessione è effettuata tra dispositivi posti in visibilità reciproca ad una distanza di 1 o 2 metri e permette una bit rate di 4Mbps. Esiste anche una implementazione seriale di IrDA, nota con il nome di SIR (Serial InfraRed), che garantisce una data rate massima di 155 Kbps.
I dispositivi che utilizzano questo standard comunicano utilizzando dei LED (Light Emission Diode) alla lunghezza d'onda di 875nm. Se da un lato questa tecnologia garantisce una certa sicurezza della trasmissione è anche vero che richiede di avere i dispositivi particolarmente vicini e fermi: l'eventuale frapposizione di un oggetto o lo spostamento di un dispositivo potrebbe comportare la caduta della connessione.
Lo standard Bluetooth verrà presentato in seguito.

1.2.3 **Wireless Local Area Network**

Le WLANs hanno un raggio di comunicazione tipico di un singolo palazzo, cioè compreso tra 100 e 500 metri. Troviamo nelle WLAN gli

stessi requisiti delle tradizionali wired LANs, come la completa connessione fra le stazioni che ne fanno parte e la capacità di inviare messaggi broadcast. Trattandosi di reti wireless, però, le WLAN devono affrontare alcuni problemi specifici di questo ambiente, come la sicurezza delle trasmissioni via etere, il consumo energetico, la mobilità dei nodi e la limitata larghezza di banda.
Due differenti approcci possono essere seguiti per l'implementazione di una wireless LAN: un approccio basato su infrastrutture o un approccio basato su reti ad hoc. L'architettura basata su infrastruttura prevede l'esistenza di un controllore centralizzato, spesso chiamato Access Piont. L'Access Point solitamente è connesso con la rete fissa, fornendo in questo modo l'accesso ad internet ai dispositivi mobili. Una rete ad hoc, invece, è una rete peer-to-peer formata da nodi mobili che si trovano all'interno dei reciproci raggi di trasmissione che dinamicamente si configurano per formare una rete temporanea. La configurazione ad hoc non richiede la presenza di un controller fisso, ma un controller è dinamicamente eletto tra tutti i nodi partecipanti alla comunicazione. Lo standard più diffuso per questo tipo di reti è l'IEEE 802.11, vedi paragrafo 1.4.1.

1.2.4 **Wireless Wide Area Network**

Le WWAN (Wireless Wide Area Network) sono le reti wireless dal range più ampio oggi disponibile, e vengono nella maggior parte dei casi installate nell'infrastruttura della fonia cellulare, sebbene offrano anche la possibilità di trasmettere dati. Le WWAN sono estese su vaste aree geografiche e hanno un raggio di trasmissione dell'ordine dei km, tipicamente compreso tra 1,5 e 8 Km. Nascono dall'esigenza di collegare utenti che si trovano a grandi distanze. Questa tipologia di rete si sta diffondendo anche come strumento per collegare diverse LAN disperse su territori lontani. Le soluzioni WWAN, che si basano su un'infrastruttura a rete cellulare, o su trasmissione satellitare, rappresentano il futuro della comunicazione dei dati di ogni giorno. Esse sono destinate a diventare sempre più importanti dato che sempre più persone hanno la necessità di accedere ai dati e scambiarli per uso personale o professionale senza essere legati ad un luogo specifico. Le Wireless Wide Area Network forniscono accesso alle informazioni *anytime & anywhere* in presenza di copertura di rete cellulare.

Le WWANs sono circondate da una varietà di standard. Lo standard Advanced Mobile Phone Systems (AMPS), nato negli Stati Uniti, permette la trasmissione della voce tra cellulari Tacs. Opera su una banda di frequenza di 800 MHz. E' basato su tecnologie analogiche e viene considerato uno satandard di prima generazione (1G).
Gli standard di seconda generazione (2G), basati su tecnologia digitale, sono stati progettati principalmente per servizi telefonici. Tra gli esempi di questa categoria troviamo il Global System for Mobile (GSM) in Europa e Personal Digital Comunication (PDC) in Giappone. La data rate prevista per le tecnologie di seconda generazione non superava 9.6 Kbps.
Gli standard 2,5G, come General Packet Radio System (GPRS) e EDGE (evoluzione di GSM), nascono dalla necessità di fornire sia i servizi telefonici che quelli per lo scambio di dati; consentono una data rate di 348 Kbps.
La telefonia di terza generazione (3G) si propone di fornire trasmissioni con data rate che raggiungono i 2Mbps al fine di supportare vari servizi multimediali, quali videotelefonate, scambio di brevi filmati, etc. Uno standard di questo tipo è l'UMTS (Universal Mobile Telecommunication System) introdotto dalla International Telecommunications Union (ITU).
L'obiettivo di UMTS è quello di andare ad estendere la capacità in termini di banda delle reti cellulari e cercare di arricchire in modo significativo la tipologia e la qualità dei servizi agli utenti. Mentre la maggior parte dei sistemi di comunicazione cellulare utilizza tecnologie switched, UMTS ha anche il supporto per le comunicazioni basate sui pacchetti che risulta sicuramente più adatto per gestire il traffico dati. Le frequenze di trasmissione utilizzate da UMTS sono comprese fra 1,9 Ghz e i 2,2 Ghz mentre lo standard GSM trasmette a 900 Mhz e a 1800 Mhz. UMTS utilizza il Direct-Sequence Code Division Multiple Access (DS-CDMA) con una larghezza di banda di circa 5 MHz. Questo è spesso noto come Wideband CDMA (WCDMA) [2]. La tecnologia di trasmissione di UMTS (UTRAN, UMTS Terrestrial Radio Access Network) consente una data rate minima di 144 Kbps per le applicazioni ad alta mobilità totale estesa a tutti gli ambienti, una data rate pari a 384 Kbps per applicazioni a mobilità parziale, e una massima data rate di 2 Mbps per le applicazioni a bassa mobilità. UTRAN prevede due modi di lavorare, Frequency Division Duplex (FDD) e Time Division Duplex (TDD). In FDD le trasmissioni uplink e downlink usano due frequenze radio separate. In TDD le

trasmissioni uplink e downlink usano la stessa frequenza a intervalli di tempo sincronizzati.

1.3 **Modelli di reti mobili**

Verranno descritti brevemente tre modelli di reti mobili [3] in ordine crescente di flessibilità e in ordine decrescente in termini di contenuto di infrastrutture fisse.

1.3.1 **Cellularar Network Model**

Una cellular network copre una determinata area composta da celle eventualmente sovrapposte. Ogni cella ha una fissa base station (BS). Le base stations sono connesse tra di loro da una rete wireline. I nodi mobili (noti anche come mobile hosts o MHs) si possono muovere da una cella all'altra. Un MH comunica con gli altri nodi della rete attraverso la base station della cella in cui si trova. Per effettuare ciò, il MH deve stabilire una connessione wireless con la sua base station. Se il partner della comunicazione è presente nella stessa cella, la base station consegna il messaggio attraverso un altro link wireless stabilito con il ricevente. Se il partner della comunicazione si trova in un'altra cella, la base station inoltra il messaggio attraverso la wireline backbone alla base station presente nella cella del ricevente. La rete backbone è anche connessa alla rete telefonica.
In una cellular network possiamo assumere che nella rete wireline backbone non si verifichino mai disconnessioni e che tutte le base station siano funzionanti. Finché due MHs sono funzionanti e situati nell'area di copertura, essi possono comunicare gli uni con gli altri. Dal punto di vista di un grafo, le base stations sono i nodi interni del grafo e gli MHs sono le foglie. Nel tempo l'unico cambiamento nel grafo riguarda le interconnessioni tra i nodi foglia e i nodi interni. Il sottografo che comprende i nodi interni rimane invariato.

1.3.2 **Virtual Cellularar Network Model**

Una virtual cellular network (VCN) è simile alla cellular network, eccetto per una grande differenza: le base stations di una VCN sono mobili. La comunicazione tra le base stations avviene anche attraverso link wireless. La mobile station continua a coordinare la comunicazione dei MHs che si trovano nella sua prossimità. Tuttavia, a differenza delle virtual cellular, il grafo delle base stations può cambiare nel tempo a causa della loro mobilità. Come conseguenza, la maggior parte degli algoritmi distribuiti che sono utilizzati nelle virtual cellular cessano di funzionare correttamente. Le VCNs possono essere sfruttate per interventi militari in territorio nemico dove le base stations mobili possono essere installati su carri armati o autocarri e i soldati possono trasportare i mobile hosts.
Nelle VCN si assume che le base stations mobili possano sempre comunicare l'una con l'altra attraverso un cammino formato esclusivamente da base stations. Ogni MH è connesso a qualche base stations mobile. Comunque, il sottografo che comprende i nodi interni (le base stations mobili) cambia nel tempo.

1.3.3 **Ad-hoc Network Model**

In una rete ad-hoc tutti i nodi sono simili e tutti sono mobili. Non ci sono base station che coordinano le attività di un sottoinsieme di nodi. Perciò tutti i nodi devono prendere le decisioni collettivamente. A causa della mobilità, i vicini di un nodo cambiano con il tempo. Poiché la mobilità dei nodi non può essere predetta, i cambiamenti della topologia della rete nel tempo sono arbitrari. Le reti ad-hoc sono adatte per le missioni militari, per le operazioni di soccorso in situazioni di emergenza, etc.
In una rete ad-hoc due nodi mobili condividono un link se sono all'interno del reciproco raggio di trasmissione wireless. Assumendo che un cammino single o multiple hop esiste per ogni coppia di nodi mobili, questo cammino può cambiare nel tempo a causa del movimento dei nodi.

1.4 Tecnologie per le wireless Lan

Il successo di una tecnologia di rete è connesso con lo sviluppo dei prodotti di rete ad un prezzo competitivo. Il maggiore fattore per ottenere questo risultato è la disponibilità di un appropriato standard di rete. Attualmente, due principali standard per le reti wireless ad hoc stanno emergendo: lo standard IEEE 802.11 per le WLANs e le specifiche Bluetooth per le comunicazioni wireless a corto raggio.

1.4.1 IEEE 802.11

L'IEEE (Institute of Electrical and Electronic Engineers) nel 1997 ha divulgato il primo standard di riferimento per le reti wireless: l'IEEE 802.11 che dettava le specifiche a livello fisico e data link per l'implementazione di una rete LAN wireless. Tale standard consentiva una date rate di 1 o 2 Mbps usando la tecnologia basata su onde radio nella banda 2.4 GHz o sugli infrarossi. La limitata banda ne determinò uno scarso successo e diffusione.
La sua evoluzione del 1999 prende il nome di IEEE 802.11b ed è diventato il nuovo standard dominante. L'IEEE 802.11b (noto anche come Wi-Fi, Wireless Fidelity) infatti consente una data rate fino a 11 Mbps. Questo standard consente collegamenti radio in un raggio di circa 250 metri.
Esiste inoltre uno standard più recente, lo standard 802.11a, che stabilisce una velocità massima di trasferimento dei dati di 54 Mbps su una banda di frequenza di 5 GHz.
Lo standard 802.11b in particolare definisce un insieme di specifiche per il livello fisico e per il livello MAC (Medium Access Control).
Infatti poiché il canale radio è un mezzo condiviso, è necessario un meccanismo di controllo di accesso al mezzo per evitare interferenze nel caso in cui più nodi tentino di accedere contemporaneamente al canale.
Il livello MAC descritto nell'802.11b prevede due differenti metodi d'accesso, *Distribuited Coordination Function* (DCF) e *Point Coordination Function* (PCF), ma solo il primo può essere utilizzato per le reti ad hoc.
Il protocollo DCF usa il meccanismo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*), tecnica di accesso multiplo con rilevamento della portante e prevenzione delle collisioni.

Prima di iniziare a trasmettere una stazione controlla il canale per verificare se un'altra stazione sta trasmettendo. Se il mezzo risulta libero per un certo intervallo di tempo, detto *Distribuited InterFrame Space* (DIFS), la stazione inizia la sua trasmissione. Il pacchetto trasmesso contiene la lunghezza della trasmissione che si sta per effettuare. Ogni stazione attiva memorizza questa informazione in un campo chiamato *Network Allocation Vector* (NAV) e quindi sa per quando tempo il canale rimarrà occupato. La stazione che riceve il pacchetto con i dati aspetta per un intervallo di tempo detto *Short InterFrame Space* (SIFS), minore del DIFS, dopodichè invia un frame di conferma ricezione (ACK). Nel caso in cui due o più stazioni inizino a trasmettere nello stesso istante, si possono verificare collisioni. L'ack non viene trasmesso se il pacchetto è stato corrotto o è andato perduto a causa di collisioni. Nel caso in cui non si ricevano gli ACK relativi ai frame inviati, si presume che questi siano andati perduti, così vengono ritrasmessi.
Viene usato un algoritmo di controllo della ridondanza ciclica (CRC, Cyclic Redundancy Check) per rilevare errori nelle trasmissioni. Nel caso in cui sia rilevato un errore, sia esso dovuto ad una collisione oppure ad un errore di trasmissione, il canale rimane inattivo per un intervallo di tempo detto *Extended InterFrame Space* (EIFS), al termine del quale le stazioni tentano la ritrasmissione dell'ultimo pacchetto attraverso un meccanismo di backoff. Questo meccanismo ha il compito di diminuire le probabilità di collisioni garantendo la diffusione del tempo di trasmissione
Quando una stazione pronta alla spedizione di un pacchetto si accorge che il canale è occupato, rimanda tale operazione fino la fine della trasmissione in progresso. Una volta libero il canale, la stazione inizia un conteggio, detto *backoff timer*, selezionando un intervallo casuale in cui stabilire l'inizio della trasmissione. Il *backoff timer* viene decrementato finché il canale è inattivo, mentre quando viene rilevata una trasmissione viene stoppato e riattivato quando il canale risulta inattivo per un tempo superiore al DIFS. La stazione inizia a trasmettere quando il *backoff timer* viene azzerato.
Lo standard IEEE 802.11 prevede inoltre l'utilizzo della tecnologia wep (wired equivalent protocol) per garantire sicurezza nello scambio di dati
Le modalità di funzionamento previste dallo standard IEEE 802.11 sono due: la modalità infrastruttura e la modalità ad–hoc.

1.4.1.1 **Infrastructure mode**

Nella modalità infrastructure è prevista la presenza di una infrastruttura di supporto per la comunicazione, detta Access Point (AP), a cui tutti i clients cioè i wireless terminal (WT), si collegano per comunicare tra di loro.
L'AP coordina la trasmissione tra i clients e, comunemente svolge anche la funzione di ponte tra LAN wireless, eventuali reti fisse e la rete telefonica pubblica (quindi Internet).
Tutto il traffico radio dei dati è da e verso l'Access Point sia che quest'ultimo sia connesso al Network cablato o meno.
Questa configurazione è chiamata Indipendent *Basic Service Set* (BSS), vedi figura 1.2.

Un *Extended Service Set* (ESS) è una configurazione che prevede due o più BSSs in una singola sotto-rete: più *access point* comunicano tra loro, consentendo una ottimizzazione del traffico tra le stazioni.

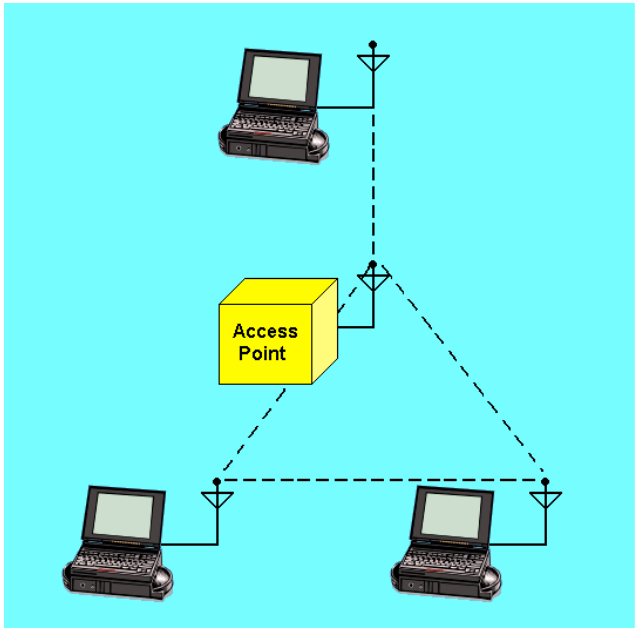


Figura 2.2 Infrastructure BSS.

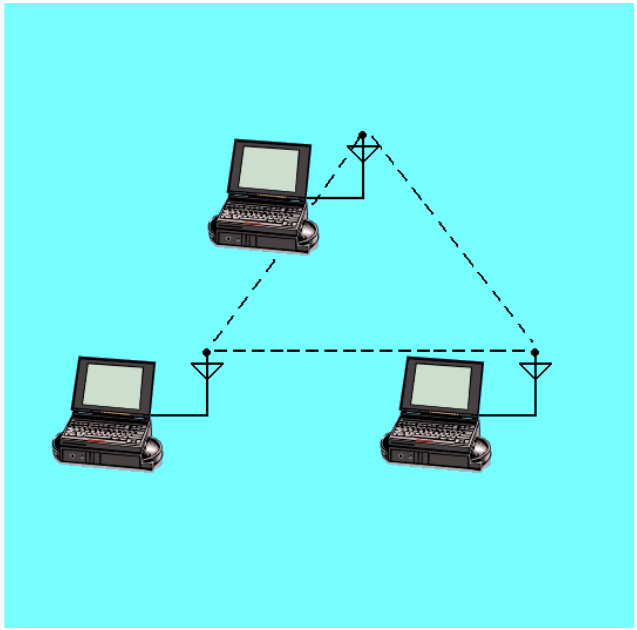


Figura 3.3 Independent BSS.

1.4.1.2 **Ad Hoc mode**

Esiste un'altra tipologia di reti wireless che non prevede l'utilizzo di infrastrutture di supporto per la comunicazione. Questa tipologia viene

definita Ad-Hoc. Le reti Ad-Hoc sono formate da un insieme nodi mobili, PDA o laptop, che comunicano fra loro con link wireless. Questi nodi possono liberamente e dinamicamente organizzarsi in una arbitraria e temporanea rete permettendo agli utenti di collaborare in modo improvvisato ovunque essi si trovino. Essendo formate da nodi mobili le reti Ad-Hoc sono note anche con il nome di MANET (Mobile Ad-hoc NETwork) [4].
Le MANET si basano sulla comunicazione cooperativa tra nodi paritetici e, senza la necessità di punti di accesso fissi, permettono la formazione di configurazioni dinamiche e sempre in mutamento, ad-hoc appunto. Ogni nodo non opera più solo come host, ma anche come router, provvedendo ad instradare quei pacchetti che, a causa del limitato range di trasmissione, non possono essere trasmessi direttamente al destinatario.
E' interessante notare che le reti ad hoc non sono preconfigurate ma si formano per la sola presenza dei vari dispositivi in un dato territorio. Ogni dispositivo (un telefono cellulare, un palmare, un computer, ma anche un elettrodomestico o una automobile) si configura come un nodo della rete, capace di far rimbalzare automaticamente le informazioni verso un altro nodo che si trova più vicino alla destinazione prescelta.
Per questo motivo le MANET sono reti multi hop, letteralmente multi-salto: questa tecnologia non si limita a far comunicare direttamente due nodi che si trovano a breve distanza ma è capace di far rimbalzare le informazioni attraverso diversi nodi per inoltrarle a un destinatario finale che può essere anche molto lontano. I protocolli di routing ad hoc prevedono quindi una partecipazione di tutti i nodi sia alla fase in cui un terminale cerca di "scoprire" percorsi nella rete, per stabilire connessioni (route discovery), che a quella di trasmissione vera e propria.
In pratica, ogni nodo utilizza gli altri dispositivi presenti nell'area per far arrivare le informazioni al destinatario; se uno dei nodi "cade", perché ad esempio gli si sono scaricate le batterie, il proprietario ha deciso di spegnerlo o si è spostato in una zona troppo lontana, la rete è capace di riconfigurarsi e trovare comunque una via per far comunicare tra loro i vari nodi.
Le reti MANET mantengono tutti i tipici problemi delle reti wireless come l'ottimizzazione della banda, la necessità di limitare i consumi energetici e, in alcuni casi, la scarsa capacità di calcolo dei dispositivi, più altri dovuti alle frequenti disconnessioni, ai continui cambiamenti della topologia della rete e al routing multi hop [5].

Le reti ad hoc sono note anche come Independent Basic Service Set Network (IBSS Network), vedi figura 1.3. La tipologia Ad-Hoc è quella che è stata presa come riferimento per il progetto di tesi.

1.4.2 **Bluetooth**

Lo standard Bluetooth consente collegamenti radio su brevi distanze (10 m), ed è stato progettato per essere economico, semplice da usare, facilmente integrabile e soprattutto a basso consumo. Le specifiche per il Bluetooth sono rilasciate dal Bluetooth Special Interest Group (SIG) [6]. Il gruppo è formato dalle maggiori aziende leader nel campo della telecomunicazione, del computering e del networking, unite per sviluppare uno standard per l'interconnessione via radio. Inoltre, il gruppo di lavoro IEEE 802.15.1 per le wireless personal area network ha approvato il primo standard per le WPAN che deriva dalle specifiche Bluetooth.
Il dominio applicativo di tale tecnologia risiede nella comunicazione e nell'interazione tra dispositivi eterogenei come telefoni cellulari, PDA, notebook, stampanti, tv, elettrodomestici, senza l'utilizzo di access point ma tramite onde radio emesse da questi dispositivi. In una rete Bluetooth solo una stazione ha il ruolo di master, mentre tutte le altre hanno il ruolo di slave. Il master decide quale slave ha accesso al canale. Più precisamente, uno slave è autorizzato ad inviare un singolo pacchetto solo se ha ricevuto un messaggio di polling dal master. I dispositivi che condividono lo stesso canale (cioè che sono sincronizzati dallo stesso master) formano una piconet. Una piconet consente una bit rate di 1 Mbps; questo valore rappresenta la capacità del canale incluso l'overhaed introdotto dai protocolli adottati e dallo schema a pooling. Una piconet può contenere contemporaneamente una stazione master e fino a sette stazioni slave attive, cioè che partecipano allo scambio dei dati.
La piconet costituisce una rete ad hoc Bluetooth single-hop. Reti Bluetooth multi-hop sono ottenute da l'interconnessione di più piconet: piconets indipendenti che coprono aree sovrapposte possono formare una scatternet. Una scatternet si forma quando un dispositivo (che funge da slave) è attivo in più di una piconet allo stesso tempo. Il traffico tra le picontes che formano la scatternet è distribuito attraverso gli slave comuni. Uno slave può comunicare con le differenti piconets cui appartiene in tempi differenti. Ciò significa che, per ogni istante, una stazione può

trasmettere solo nella piconet alla quale è sincronizzata. Per trasmettere in un'altra piconet deve cambiare i parametri di sincronizzazione. Gli algoritmi di formazione delle scatternet e gli algoritmi di scheduling del traffico tra le varie piconets sono problemi che devono essere ancora risolti [7,8].
I dispositivi Bluetooth, assicurando la compatibilità con lo standard IEE 802.11, operano nella banda dei 2,4 GHz. La tecnica di modulazione utilizzata da Bluetooth per la trasmissione dei dati è nota come *Frequency Hopping Spread Spectrum* (FHSS). I dati trasmessi vengono modulati in modo casuale 1600 volte al secondo su una delle 79 bande di frequenza disponibili. Ciò garantisce la sicurezza in Bluetooth contro eventuali tentativi di intercettazione, evitando allo stesso tempo interferenze con altri dispositivi che utilizzano la stessa banda di frequenza.

1.5 **Protocolli di routing**

La natura dinamica delle reti MANET derivante dai frequenti e imprevedibili cambiamenti della topologia della rete unita alla mancanza di infrastrutture rende particolarmente difficile l'instradamento dei pacchetti attraverso i nodi mobili. L'assenza di infrastrutture obbliga ogni nodo ad operare non più solo come host, ma anche come router, provvedendo ad instradare quei pacchetti che, a causa del limitato range di trasmissione, non possono essere trasmessi direttamente al destinatario. Ciò deriva dal fatto che ogni nodo della rete è capace di comunicare solo con i nodi che si trovano nel proprio raggio di trasmissione R. Se un nodo sorgente S vuole inviare un pacchetto a un nodo destinatario D, la cui distanza è più grande di R, il pacchetto dovrà necessariamente attraversare dei nodi intermedi. Per questo motivo le MANET appartengono alla categoria delle reti wireless multi-hop.
Il compito del routing è di trovare un percorso, nel caso in cui non esista un collegamento diretto, tra la sorgente e la destinazione che passi attraverso altri nodi e di inoltrare i pacchetti (forwarding) attraverso il cammino trovato. Il percorso è calcolato da algoritmi che operano a livello di rete e che solitamente includono tecniche per il suo mantenimento. Ci sono due principali tecniche per realizzare il forwarding. La prima prevede che ogni nodo memorizzi in un Routing Table (RT) locale i nodi successivi nei percorsi verso gli altri nodi della rete. La seconda, chiamata

source routing, prevede ogni pacchetto memorizzi nell'header il percorso completo.
I protocolli di routing per una rete Mobile Ad Hoc devono adattarsi ai frequenti cambiamenti della rete al fine di fornire il corretto percorso che un pacchetto deve seguire per raggiungere il suo destinatario. Ciò porta inevitabilmente un alto traffico di segnalazione, in contrasto con la necessità delle reti wireless di minimizzare l'utilizzo delle risorse di comunicazione.
Nei casi in cui la topologia della rete cambia con estrema velocità, gli algoritmi di routing non riescono ad aggiornare i cammini con tempestività, e il routing può essere realizzato solamente attraverso la tecnica di flooding [9,10].

1.5.1 **Routing unicast**

Lo scopo dei protocolli di routing unicast è coordinare i nodi della rete per permettere la comunicazione point-to-point tra due nodi, quando è necessario inviare un pacchetto da un nodo sorgente ad un nodo destinatario.
Troviamo molti protocolli di ruting di tipo unicast alcuni creati appositamente per ambienti MANET, altri sono "estensioni"(adattamenti) dei protocolli convenzionali delle reti wired. Non essendo definito uno standard, possiamo classificare quelli esistenti in tre categorie: proattivi, reattivi e ibridi.

1.5.1.1 Protocolli Proattivi

La caratteristica principale dei protocolli proattivi è quella di cercare di mantenere costantemente informazioni sui percorsi di routing tra ogni coppia di nodi presenti nella rete propagando aggiornamenti ad intervalli regolari. Tutti i cammini possibili sono calcolati indipendentemente dal loro effettivo uso.
Poiché queste informazioni sono generalmente memorizzate dai nodi nelle Routing Table, questi protocolli vengono chiamati anche protocolli Table-Driven.

Il vero vantaggio dei protocolli di questa famiglia è che il percorso verso la destinazione è disponibile in ogni momento e non si verificano ritardi quando l'applicazione deve inviare messaggi.
Tra i protocolli di routing proattivi più rappresentativi troviamo:
Destination-Sequenced Distance-Vector (DSDV), Optimized Link State Routing (OLSR).
Il protocollo **Destination-Sequenced Distance-Vector (DSDV)** descritto in [11], è di tipo Table-Driven. Ogni nodo mobile della rete mantiene una tabella di routing nella quale sono memorizzate per ogni possibile destinazione della rete il next-hop del cammino, il numero di hops di distanza e un numero sequenziale assegnato dal nodo destinazione. Il numero sequenziale permette ai nodi mobili di distinguere il cammino vecchio da quello nuovo e di evitare i routing loops e il fenomeno di counting-to-infinity presenti nei protocolli Distance Vector classici. Gli aggiornamenti delle tabelle di routing sono periodicamente diffuse attraverso la rete al fine di mantenerne la consistenza. Per aiutare a diminuire la quantità di traffico nella rete che tali aggiornamenti possono generare, vengono usati due tipi di pacchetti. Il primo, noto come *full dump*, contiene tutte le informazioni di routing disponibili e può richiedere più NPDUs (network protocol data units). Nei momenti in cui la topologia della rete subisce variazioni occasionali, questi pacchetti sono trasmessi raramente. Pacchetti *incremental*, di dimensioni più piccole, sono usati per trasmettere solo le informazioni che sono cambiate dall'ultimo full dump. Questi pacchetti hanno la dimensione standard di un NPDU e diminuiscono il traffico generato. La notizia di una destinazione irraggiungibile è immediatamente diffusa dal nodo che rileva il fallimento. Inoltre, alla tabella di routing, per evitare il diffondersi di cammini inaffidabili, viene associato un *time setting* , ovvero un intervallo di tempo trascorso il quale il cammino può essere considerato stabile. Se in tale periodo dopo il primo aggiornamento si ricevono altre informazioni che lo confermano, il nuovo cammino viene propagato, altrimenti viene scartato.

1.5.1.2 Protocolli Reattivi

Questa categoria di protocolli prevede di calcolare i percorsi di routing solo al momento del bisogno, ovvero quando è necessaria una trasmissione di dati, e tali informazioni sono mantenute soltanto fin

quando sono necessarie. Perciò questo approccio non ha bisogno di tabelle di routing su ogni nodo e in assenza di traffico l'attività di routing è totalmente assente anche se la rete varia la propria topologia..
Questi protocolli vengono chiamati anche protocolli on-demand.
Un protocollo reattivo è caratterizzato dalle seguenti fasi: Route Discovery, Route Maintenance e Route Deletion. La Route Discovery serve per scoprire il cammino verso la destinazione. A tale scopo la sorgente diffonde un pacchetto di query (RREQ); quando il pacchetto arriva alla destinazione viene trasmessa alla sorgente la risposta (RREP). La Route Maintenance provvede ad aggiornare il cammino trovato nella route discovery. La Route Deletion libera le risorse occupate sui nodi intermedi quando un cammino non viene più utilizzato.
Tra i protocolli di routing reattivi più rappresentativi troviamo: Dynamic Source Routing (DSR), Ad hoc On Demand Distance Vector (AODV), Temporally Ordered Routing Algorithm (TORA).
Il **DSR** [12] è un protocollo di routing on-demand basato sul concetto di source routing: i pacchetti vengono inoltrati verso il nodo destinazione in base alle informazioni di instradamento incluse nel pacchetto stesso dal nodo sorgente. Tutti i nodi mantengono nelle caches i percorsi noti verso la destinazione. Le entry nelle cache sono continuamente aggiornate quando viene trovato un nuovo cammino. Come tutti i protocolli reattivi DSR è composto dalle fasi route discovery e route maintenance. Quando un nodo ha un pacchetto da inviare, prima consulta la sua cache per vedere se ha già il cammino verso la destinazione. Se il nodo non ha il percorso inizia la fase di discovery trasmettendo in broadcast il pacchetto RREQ. Questo pacchetto contiene l'indirizzo della destinazione, insieme con l'indirizzo della sorgente ed un numero identificativo. Ogni nodo che riceve il pacchetto controlla se conosce un percorso verso la destinazione. Se no, aggiunge il suo indirizzo nel campo route record del pacchetto e poi rinvia in broadcast il pacchetto. La RREP è generata quando la RREQ raggiunge la destinazione o un nodo intermedio che conosce il cammino verso la destinazione. Nel momento in cui il pacchetto raggiunge la destinazione o il nodo intermedio, contiene nel campo route record la sequenza di hops che formano il cammino. Se il nodo che genera la RREP è il destinatario, questo prende il campo route della RREQ e lo mette nella RREP. Se il nodo che genera la RREP è un nodo intermedio, questo appende il cammino che ha nella cache al campo route della RREQ e poi genera la RREP. Per inviare la RREP , il nodo che risponde deve avere il

cammino verso la sorgente. Se ha tale cammino nella sua cache può usarlo. Altrimenti, se i link sono bidirezionali, il nodo si costruisce il cammino inverso a partire da quello che si trova nella RREQ. Se i link non sono bidirezionali, il nodo può iniziare la sua route discovery mettendo in piggyback la RREP nella RREQ. La route mainteniance è realizzata attraverso l'uso di pacchetti di errore e di ack. I pacchetti di errore (RERR) sono generati su un nodo quando il livello data link incontra un problema di trasmissione. Quando un RERR è ricevuto, l'hop che ha l'errore viene rimosso dalla cache dei cammini e tutti i cammini che contengono quell'hop sono troncati in quel punto. Oltre ai RERR sono usati pacchetti di ack per verificare il corretto funzionamento dei link.

1.5.1.3 Protocolli Ibridi

I protocolli ibridi combinano gli approcci proattivi e reattivi cercando di mettere insieme i vantaggi di entrambe. In pratica, il calcolo del percorso di routing avviene in parte in modo proattivo e parte in modo reattivo: i percorsi verso i nodi vicini vengono calcolati in anticipo come previsto nei protocolli proattivi, mentre i percorsi verso nodi lontani vengono calcolati on demand sfruttando le funzionalità reattive.
Tra gli esempi di questo tipo di protocollo troviamo Zone-Based Hierarchical Link State Routing Protocol (ZRP) e Location Aided Protocol (LAR).
ZRP [13] si basa sul concetto di zona; una zona Z(k,n) di raggio k centrata sul nodo n è definita come l'insieme di nodi distanti da n non più di k-hops:

$$Z(k,n) = \{i \mid H(n,i) <= k \},$$

dove H(i,j) è la distanza in numero di hop tra il nodo i e il nodo j. Il nodo n è chiamato il nodo centrale della zona di routing, mentre i nodi per i quali la distanza da n è uguale a k sono chiamati nodi periferici.
L'architettura del protocollo è organizzata in quattro componenti principali: l'*IntrAzone Routing Protocol* (IARP), l'*IntErzone Routing Protocol* (IERP), il *Bordercast Protocol* (BRP), il *Neighbor Discovery/Maintenance Protocol* (NDP).
L'*IntrAzone Routing Protocol* (IARP) effettua in modo proattivo il routing all'interno della zona del nodo sorgente. IARP usa il *Neighbor Discovery/Maintenance Protocol* (NDP) per rilevare i nodi vicini.

L'*IntErzone Routing Protocol* (IERP) è utilizzato per servire *on-demand* le richieste di routing verso i nodi fuori della zona. L'IERP usa una forma di flooding selettivo sfruttando la struttura sottostante generata da IARP. In pratica, il flooding è basato sull'invio di pacchetti di query solo ai nodi periferici, usando una sorta di trasmissione multicast speciale, soprannominata bordercast.
Il *Bordercast Protocol* (BRP) è un protocollo di flooding selettivo, che collabora con IARP e viene utilizzato da IERP per la ricerca del routing verso i nodi esterni alla zona.
Quando un nodo riceve il pacchetto di query, questo può rispondere alla sorgente, se la destinazione è un membro della sua zona, o effettuare il bordercast della query ai suoi nodi periferici. Il percorso verso la destinazione può essere accumulato nel pacchetto di query durante il forwarding (come per il DSR) o, per ridurre la lunghezza del pacchetto di query, nel pacchetto di controllo reply durante la fase di risposta.

1.5.2 **Multicast Routing**

Il multicast routing si occupa della trasmissione di pacchetti a un gruppo di nodi identificati da un unico indirizzo di destinazione.
Un gruppo è formato da un insieme di entità che hanno uno scopo comune. Tipicamente l'insieme dei membri di un gruppo è dinamico: in qualsiasi momento un nodo può entrare a far parte del gruppo attraverso la funzione di join o può uscirne tramite la funzione di leave. Non c'è restrizione sulla posizione o sul numero di membri in un gruppo. Una entità può appartenere contemporaneamente a più gruppi. Un nodo membro di un gruppo riceve tutti i messaggi ad esso destinati. Un nodo non deve necessariamente essere un membro del gruppo per potervi inviare un messaggio.
Gli studi per i protocolli che permettono il multicast nelle MANET sono partiti da quelli esistenti per le reti wired. Ci sono due principali approcci per il multicast routing usati nelle reti wired: quello basato sul group-share tree e quello sul source-specific tree. In entrambe i casi sono costruiti alberi che interconettono tutti i membri del gruppo multicast. I dati sono consegnati lungo i cammini dell'albero per raggiungere tutti imembri del gruppo. Nell'approccio source-specific ogni nodo mantiene un albero

verso tutti gli altri membri del gruppo. Nell'approccio group-share un singolo albero è costruito per l'intero gruppo.
Molti protocolli multicast per reti ad hoc basati sugli alberi sono stati proposti adattando quelli esistenti per le reiìti wired. Tuttavia, la topologia delle reti MANET può cambiare molto frequentemente e il mantenimento di alberi connessi può causare un eccessivo overhead. Per evitare questo è stato proposto un differente approccio basato sulle mesh. Le mesh sono più adatte per gli ambienti dinamici poiché supportano più connessioni rispetto agli alberi. Queste permettono di evitare gli inconvenienti degli alberi multicast, come le connessioni intermittenti, la concentrazione di traffico o le frequenti riconfigurazioni dell'albero. Sebbene le multicast meshs hanno un rendimento migliore degli alberi multicast nelle reti dinamiche, il meccanismo delle mesh è più incline a formare routing loops; inoltre approcci di costruzione delle mesh basati sul meccanismo di flooding producono un eccessivo overhead nelle reti di grandi dimensione.
Tra gli esempi di protocolli di routing multicast basati sulle mesh troviamo On-Demand Multicast Routing Protocol (ODMRP).
ODMRP [14] è un protocollo di routing multicast per reti mobili ad hoc basato su una struttura di distribuzione a mesh costruita attraverso il flooding. Quando un nodo ha intenzione di spedire dei dati ad un gruppo inizia a trasmettere in broadcast un pacchetto di controllo, chiamato JOIN_DATA. Quando un nodo intermedio riceve questo pacchetto, memorizza l'ID del mittente e il numero sequenziale nella sua cache per individuare eventuali messaggi duplicati. Viene effettuato l'aggiornamento della routing table con l'ID del nodo da cui è stato ricevuto il messaggio per costruire il cammino inverso verso il mittente. Se il messaggio non è un duplicato e il TTL è maggiore di zero, questo viene rinviato in broadcast.
Quando il pacchetto JOIN_DATA arriva al gruppo destinatario, questo crea una JOIN_TABLE che invia ai suoi vicini. Quando un nodo riceve la JOIN_TABLE controlla se è in tale tabella è presente il suo ID ed in questo caso, realizzando di essere un nodo intermedio tra la sorgente ed il gruppo destinatario, attiva il campo FG_FLAG, forwarding group flag. La JOIN_TABLE viene propagata estraendo le informazioni dalla routing table fino a quando non raggiunge la sorgente del JOIN_DATA attraverso il cammino scelto. Nel ricevere la JOIN_TABLE un nodo deve anche costruire la sua multicast table, aggiungendo il mittente come next hop, per poter trasmettere i futuri messaggi broadcast.

Dopo che il gruppo dei nodi forwarding è stato stabilito, la sorgente può inviare i pacchetti multicast ai riceventi e, fin quando ha messaggi da inviare, invia periodicamente pacchetti JOIN_DATA per aggiornare il gruppo dei nodi forwarding.
In ODMRP è previsto l'invio esplicito di pacchetti di controllo per effettuare le operazioni di join o leave da un gruppo. Se una sorgente multicast vuole effettuare il leave da un gruppo smette di inviare i pacchetti JOIN_DATA, dal momento che non ha più dati da inviare al gruppo. Se un membro del gruppo vuole smettere di ricevere le informazioni indirizzate a quel gruppo smette di inviare la join reply per quel gruppo. I nodi che appartengono al gruppo dei forwarding smettono di trasmettere i pacchetti ai nodi dai quali non hanno ricevuto la JOIN_TABLE entro un certo intervallo di tempo.

CAPITOLO 2

AGAPE:
Allocation and Group Aware Pervasive Environment

La crescente diffusione di dispositivi wireless e la comparsa di nuove tipologie di reti, come le Mobile Ad Hoc Networks (MANETs), ha creato nuove opportunità per gli utenti di stabilire collaborazioni impromptu. In tali scenari utenti sconosciuti che condividono interessi comuni hanno la necessità di creare gruppi on-demand e di agganciarsi/sganciarsi dai gruppi di loro interesse disponibili nella loro località. Questi servizi collaborativi in ambienti MANET, privi di infrastrutture di rete fisse e dove i membri spesso cambiano la loro posizione nella rete, sollevano complessi problemi di gestione dei gruppi.
Poiché i membri di un gruppo appaiono e scompaiono in maniera non predicibile la topologia della rete non può essere determinata staticamente, disconnessioni e partizioni di rete sono eventi comuni ed è impossibile fare assunzioni sullo stato e sulla disponibilità dei membri di un gruppo.
Inoltre i membri di un gruppo operano da dispositivi di accesso eterogenei con differenti proprietà in termini di capacità di calcolo e di memoria, dimensione dello schermo e apparato di rete.
Lo sviluppo e l'esecuzione di applicazioni in un sistema mobile distribuito di tipo Manet, può essere facilitato attraverso la realizzazione di un'infrastruttura software, detta comunemente middleware.
Possiamo definire il middleware come lo strato di software collocato tra i componenti del sistema distribuito e i componenti del sistema operativo di rete, un intermediario che fornisce agli sviluppatori un alto livello di astrazione nascondendo le complessità introdotte dalla distribuzione, come l'eterogeneità, la scalabilità, le risorse condivise.

2.1 **AGAPE**

AGAPE (**A**llocation and **G**roup **A**ware **P**ervasive **E**nvironment) è un middleware per il supporto alla gestione dei gruppi in scenari MANET.
Per permettere lo sviluppo di applicazioni collaborative in ambienti MANET, AGAPE utilizza la comunicazione message-oriened context-aware: la selezione del destinatario viene fatta in base al contesto e alla sua valutazione dinamica [15]. Il destinatario non è scelto in base al nome, ma in base al suo profilo, che include le preferenze di collaborazione e le proprietà, e in base alla locazione e alla sua posizione rispetto agli altri membri del gruppo.
La collaborazione in AGAPE è basata sul concetto di gruppo di entità: solo le entità che sono membri dello stesso gruppo possono interagire. Un gruppo è inteso come un insieme logico di entità associate insieme secondo

determinati criteri specifici dell'applicazione. Ogni gruppo è caratterizzato da un identificatore univoco GID (Group Identifier) e da un profilo di gruppo che specifica gli interessi, le preferenze e le attività del gruppo, che sono condivisi da tutti i membri del gruppo. Ogni membro del gruppo ha un identificatore personale PID (Personal Identifier) che è unico all'interno del gruppo.
Inoltre AGAPE è stato progettato per far fronte all'eterogeneità dei dispositivi usati dai clienti considerando le loro limitate capacità computazionali.
AGAPE è un middleware location-aware per la gestione dei gruppi di entità che basa la gestione del gruppo sulla posizione dei dispositivi/utenti. L'assunzione di fondo è che gli utenti situati nella stessa località della rete richiedano reciproche collaborazioni molto più frequentemente che con utenti situati in aree di reti distanti.

2.2 **Modello AGAPE**

Il supporto di rete per AGAPE è costituito da reti wireless conformi ai modelli cellular e virtual cellular [3]. In entrambe i modelli una base station (BS) fornisce la copertura della rete ai dispositivi che si trovano all'interno del raggio d'azione della rispettiva cella (vedi figura 2.1). Il raggio della cella e il numero massimo di dispositivi che possono interagire all'interno di questa dipende dalla specifica tecnologia adottata per l'implementazione della rete wireless. Una cella può anche essere collegata ad una rete fissa in modo da fornire la connessione per dispositivi fissi.
Il modello AGAPE è centrato intorno all'astrazione di dominio che rappresenta un insieme di entità raggruppate sulla base di relazioni generate da proprietà comuni. L'astrazione di dominio nasconde agli sviluppatori di applicazioni i dettagli del supporto di rete e fornisce una facile via per organizzare e mappare gruppi logici sopra l'infrastruttura fisica di rete.
Come mostra la figura 2.1, possiamo distingure due tipi di entità nel modello AGAPE: Managed Entity (ME) e Locality Manager Entity (LME).

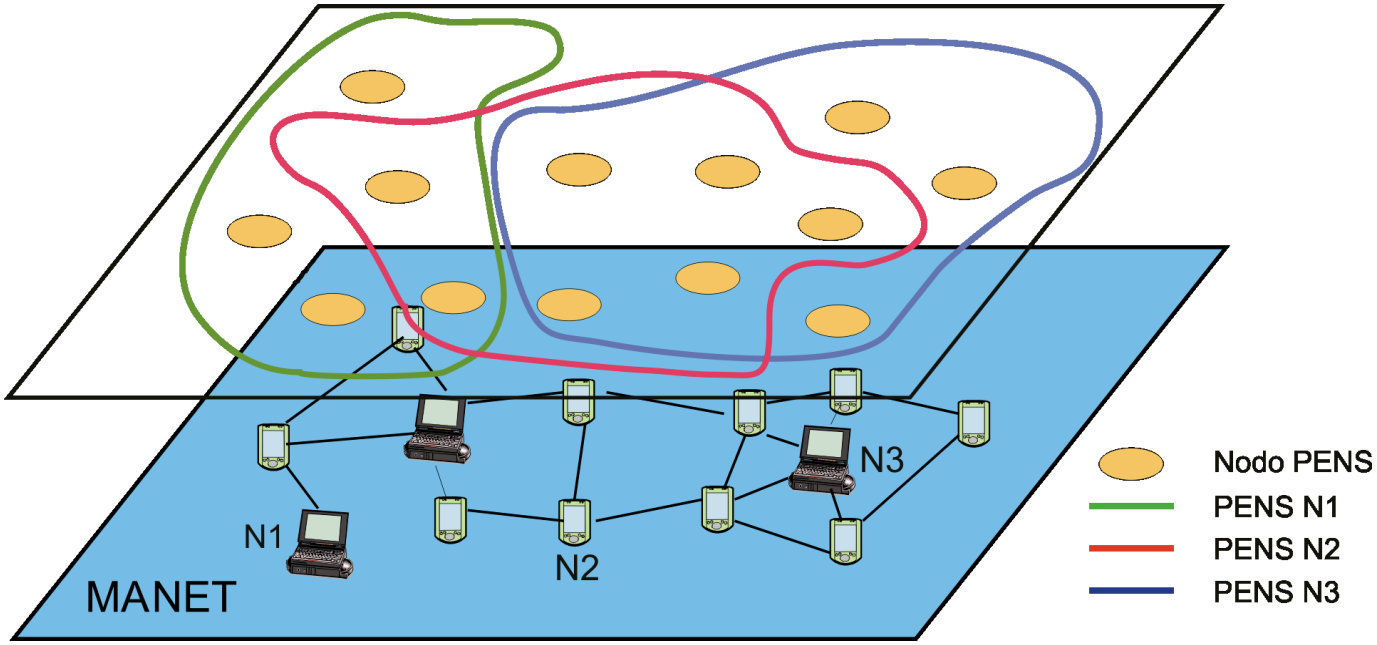


Figura 4.1 Modello di AGAPE in uno scenario wireless

Gli ME sono i membri gestiti che utilizzano i servizi di AGAPE sulla gestione dei gruppi per collaborare. In genere questo ruolo è adatto ai dispositivi con scarse capacità computazionali.
Gli LME sono i gestori dei membri del gruppo della loro località: oltre a collaborare con gli altri membri, sfruttando le elevate capacità di calcolo, per fornire le operazioni di gestione del gruppo in favore dei dispositivi ME. Gli LME favoriscono la creazione run-time di un nuovo gruppo, permettono alle entità di unirsi al gruppo e mantengono una lista aggiornata dei membri del gruppo co-locati.
Il concetto di località in AGAPE è dominante [16]: ogni LME supporta le operazioni di gestione per i membri del gruppo che si trovano nella sua località. La località di un LME è un'astrazione logica che individua un insieme di entità che sono connesse al dispositivo LME da un cammino inferiore ad una determinata soglia. Questo valore è espresso in hop e viene scelto in base al contesto applicativo. Si può notare che le entità di una località possono appartenere allo stesso o a differenti gruppi logici.
La scelta di favorire l'interazione tra membri che appartengono alla stessa località è stata suggerita dalle caratteristiche dell'ambiente MANET, dove l'impossibilità di contare su connessioni di rete stabili e la difficoltà nel consegnare messaggi attraverso percorsi di routing lunghi rende difficile realizzare l'interazione tra membri distanti. In ogni località della rete vengono disseminate periodicamente le viste dei gruppi presenti; tali viste

contengono la lista degli utenti del gruppo presenti nella località dell'LME che la gestisce.

2.2 **Architettura AGAPE**

AGAPE fornisce una serie di servizi per le applicazioni collaborative in ambienti pervasivi. Tali servizi, come mostrato nella figura 2.2, sono organizzati in due livelli logici costruiti sopra la Java Virtual Machine: il Basic Service Layer e il Group Management Layer.

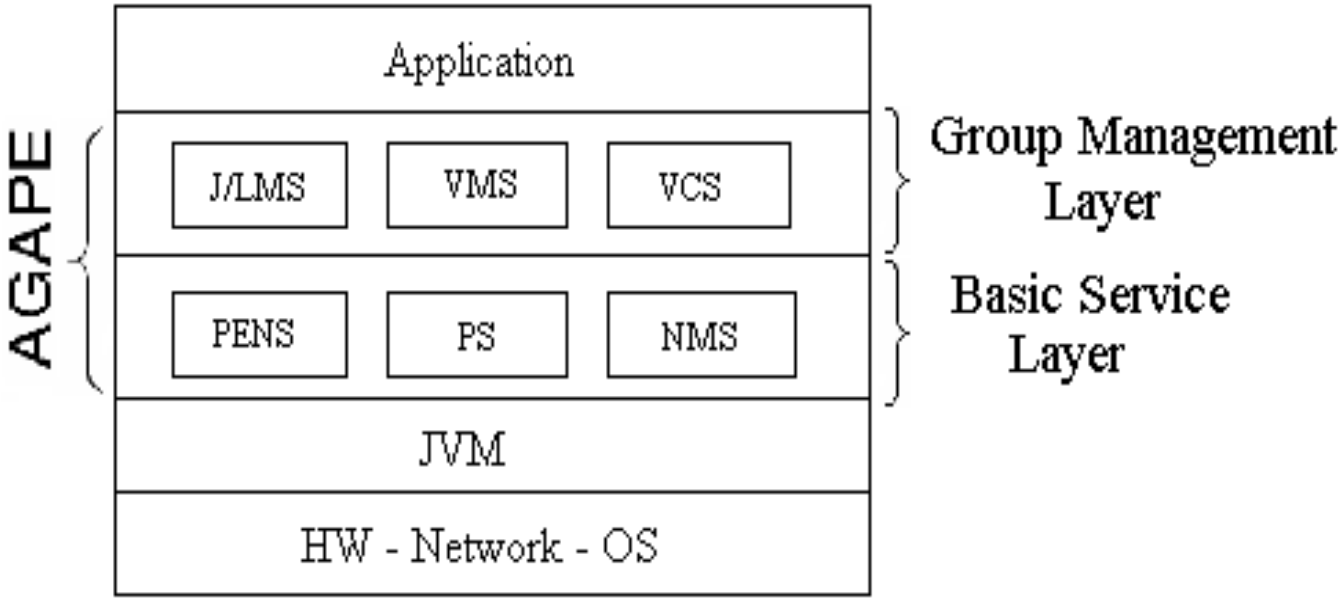


Figura 5.2 Architettura di AGAPE

2.2.1 **Basic Service Layer**

Il Basic Service Layer fornisce i servizi base per la comunicazione e realizza il servizio di Proximity usato per determinare il set di nodi mobili presenti nella prossimità di un dato nodo nella rete .
In particolare:

- Il Network Manager Service (NMS) permette ai dispositivi delle entità di AGAPE di inviare e ricevere pacchetti UDP a/dalla MANET e di realizzare la comunicazione punto-punto e punto-multipunto basata sullo scambio di messaggi. Le primitive di comunicazione che il NMS supporta sono di tipo connectionless;

quelle che realizzano la comunicazione punto-punto permettono lo scambio di messaggi tra pari della rete, mentre le primitive di comunicazione che realizzano la comunicazione punto-multipunto permettono la comunicazione tra un pari ed una molteplicità di altri pari della rete sia in modalità broadcast (verso tutti i pari) che in modalità multicast (verso un sottoinsieme o un gruppo di pari).

- Il Proximity Service (PS) permette alle entità pari, sia LME che ME, di pubblicare periodicamente i propri identificatori, in modo che gli altri nodi presenti nella località possano avvertire la sua presenza. Infatti, utilizzando le primitive fornite dal NMS, ad intervalli regolari il PS trasmette in broadcast a tutti i vicini dei beacon che includono gli identificatori GID/PID e il ruolo (ME o LME) del membro.
- Il Proximity Enabled Naming Service (PENS) mantiene una tabella aggiornata delle entità presenti nella sua località. Questo servizio infatti riceve i pacchetti trasmessi in broadcast dal PS e, in base alle informazioni ricevute, costruisce una tabella in cui associa gli identificatori GID/PID e il ruolo (LME/ME) dei membri con il loro indirizzo IP. Inoltre il PENS offre le funzionalità per generare in modo casuale gli statisticamente unici identificatori di gruppo (GIDs) e gli identificatori personali dei pari (PIDs) sfruttando un approccio di naming simile a quello proposto per gli ambienti P2P.

2.2.2 **Group Management Layer**

Il Group Management Layer fornisce i servizi richiesti per creare/dissolvere e gestire i gruppi.
In particolare:

- Il Join/Leave Manager Service (J/LMS) permette alle entità di agganciarsi/sganciarsi dai gruppi e di ridefinirsi quando cambiano le loro informazioni di profilo. Se una entità è autorizzata ad entrare in un gruppo, il J/LMS restituisce all'entità il profilo del gruppo e i suoi identificatori GID/PID.
- Il View Manager Service (VMS) permette agli LMEs di creare le viste di un gruppo e di disseminarle ad intervalli regolari nella propria località. Ogni membro del gruppo riceve la vista (context-dependent view) che contiene la lista aggiornata dei membri del gruppo che si trovano nella località dell'LME che ha inviato la vista.

In particolare, ogni vista contiene gli identificatori GID/PID e il livello batteria del dispositivo LME che l'ha generata ed è composta da una lista di entry che associa ogni riferimento di un membro del gruppo (ottenuto dal PENS) con le informazioni del profilo dell'utente/dispositivo (ottenute dal J/LMS). Il VMS si coordina con il PENS per aggiornare la vista in seguito ad arrivi, partenze o disconessioni di membri del gruppo.

- Il View Coordination Service (VCS) permette agli LMEs di distribuire o meno la vista. In questo modo, VCS aiuta a ridurre le propagazioni di viste non necessarie, per esempio, quando più LMEs appartenenti allo stesso gruppo e situati nella stessa località tentano entrambe di disseminare la vista ai membri del gruppo co-locati. In particolare, ogni volta che un LME cerca di disseminare la sua vista richiede il permesso al VCS. Il VCS confronta le entry delle viste ricevute dagli LMEs co-locati. Se le viste ricevute non includono le stesse entry, VCS permette la disseminazione della vista, mentre se le entry corrispondono il permesso di propagare la vista viene concesso solo ad un LME.

2.3 **Dettagli implementativi**

Sono state sviluppate due diverse release di AGAPE: una completa, che fornisce agli LME (caratterizzati da elevate risorse computazionali) tutti i servizi per le operazioni di gestione dei gruppi e una limitata destinata a supportare solo i servizi propri degli ME: J/LMS e VMS client side, insieme con PS, PENS e NMS.
E' importante notare che i servizi AGAPE, essendo destinati a dispositivi portatili, sono stati sviluppati cercando di ottimizzare l'uso di risorse.
A tale scopo esaminiamo alcuni brevi esempi che mettono in mostra come il J/LMS, il VMS e il VCS si comportano per evitare la trasmissione di viste non necessarie. Per ogni esempio, a cui corrisponde una possibile situazione operativa, vengono messe in rilievo le fasi di joining ad un gruppo, di propagazione delle viste nelle località presenti e di roaming. Particolare attenzione viene fatta sulla gestione delle viste nelle diverse località a sottolineare il comportamento del VMS in questi casi.

2.3.1 **Località innestate**

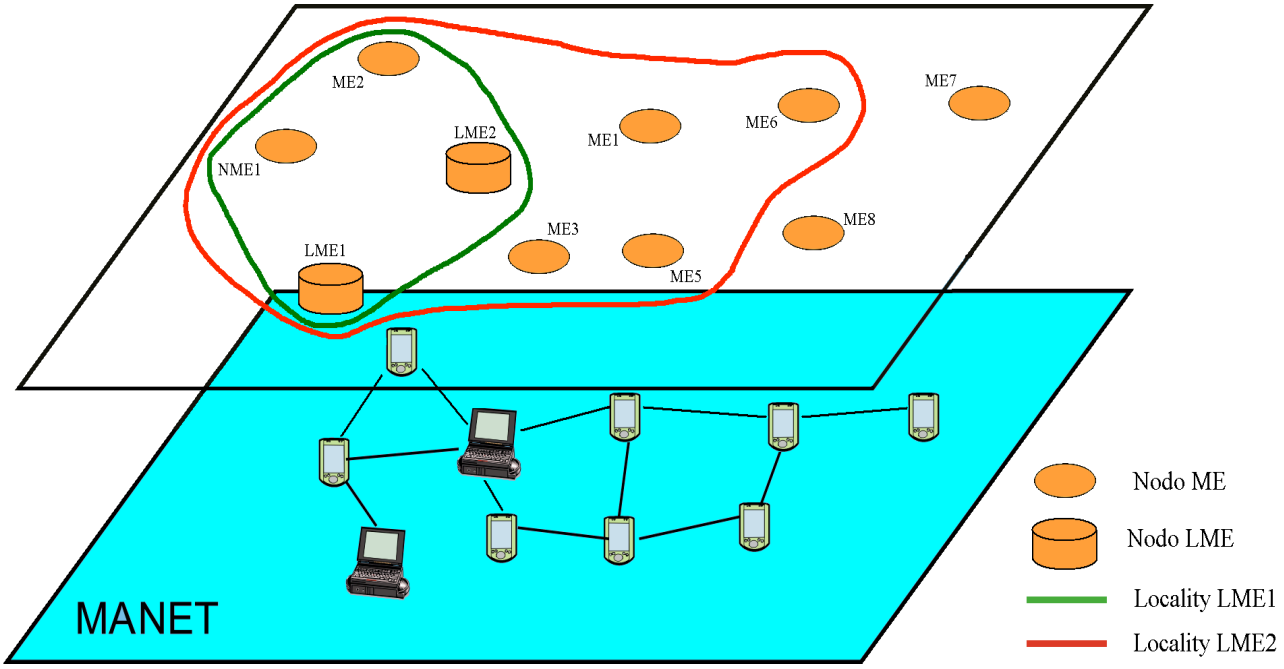


Figura 2.3 Località innestate.

La figura 2.3 mostra una situazione operativa dove due LMEs (LME1 e LME2) appartengono allo stesso gruppo, cioè hanno lo stesso GID, e definiscono due località innestate, con la località di LME1 che è contenuta nella località di LME2.

JOINING. Una entità non membro NME1 che entra nella località di LME1 può usare il PENS per ottenere la lista di tutti gli LME presenti nella località. In questo caso la lista conterrà LME1 e LME2 insieme con i loro GIDs, PIDs e IPs. Il J/LMS di NME1 riconosce che LME1 e LME2 appartengono allo stesso gruppo e può decidere di inviare la richiesta di join a uno di questi due (supponiamo LME1). Il J/LMS di LME1 riceve la richiesta di join (che contiene i profili utente e del dispositivo di NME1) e confronta se le preferenze di gruppo espresse da NME1 matchano con il profilo di gruppo. Se il confronto da esito positivo LME1 invia un messaggio di ack che contiene gli identificatori GID/PID a NME1 e include NME1 nel gruppo (inoltre si coordina con il VMS per aggiungere NME1 nella vista del gruppo).

PROPAGAZIONE DELLE VISTE. Sia LME1 che LME2 mantengono una vista aggiornata di tutti gli elementi del gruppo presenti nella loro località. A intervalli regolari entrambe le istanze VMS di LME1 e di LME2 chiedono il permesso al VCS per propagare la vista ai membri delle rispettive località. In una situazione del genere il VCS di LME1 riconosce che tutti i membri della vista sono contenuti nella vista di LME2 e quindi non permette la propagazione della vista, mentre il VMS di LME2 avrà il

permesso di disseminare la vista nella sua località. Possiamo notare che solo quando NME1 effettua il join e il VMS di LME1 include il nuovo membro nella vista, il VCS di LME1 permetterà di propagare la vista ai membri della sua località. In tale circostanza sia LME1 che LME2 trasmettono le loro viste. Poi, quando LME2 riceve la vista di LME1 si accorge del nuovo membro NME1 e, dopo aver constatato che si trova nella sua località (attraverso il PENS) lo include nella sua vista. Quindi quando LME1 riceverà la nuova vista di LME2 che include NME1, il VCS impedirà all'LME1 di propagare la vista.
ROAMING. Consideriamo il caso in cui un membro ME1 appartenente allo stesso gruppo di LME1 e di LME2 entra nella località di LME2.
L'istanza VMS di ME1 riceve la vista aggiornata inviata da LME2 e riconosce che non è incluso in quella vista. Al fine di essere incluso nella vista di LME2, l'istanza VMS di ME1 invia le sue informazioni di profilo (ottenute dal suo J/LMS) a LME2. Dopo averne ricevuto il profilo LME2 include ME1 nella sua vista. Simili considerazioni si applicano nel caso di roaming di LMEs tra località differenti.

2.3.2 **Località parzialmente sovrapposte**

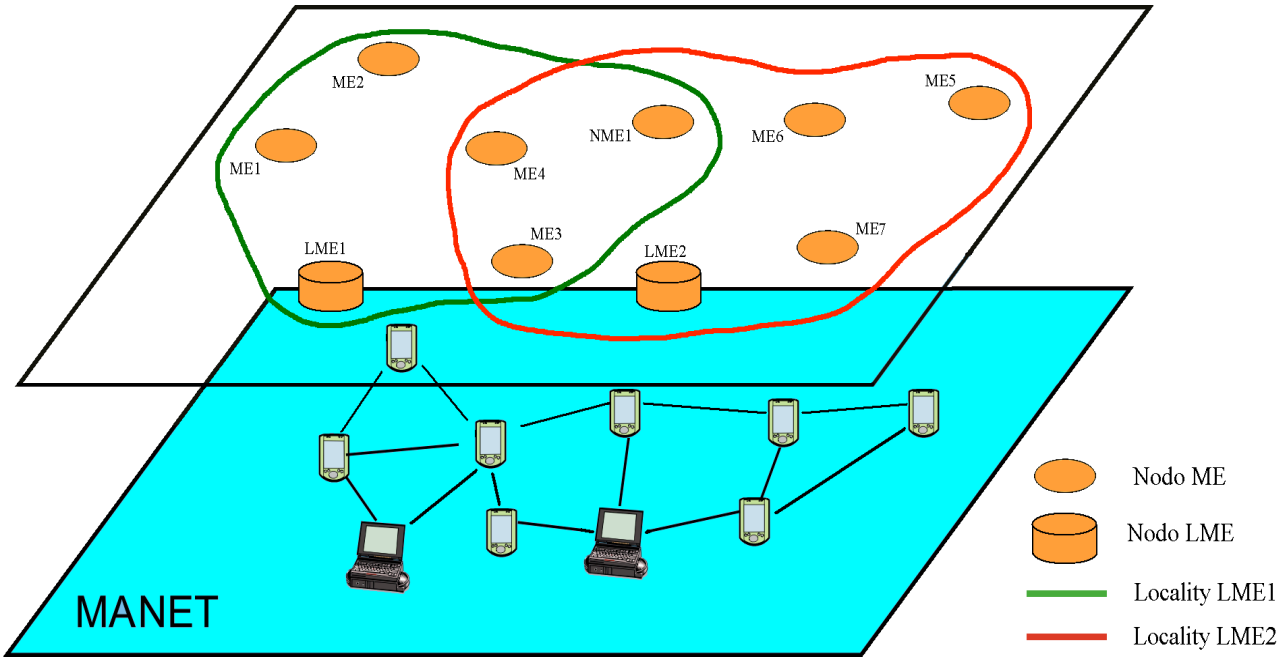
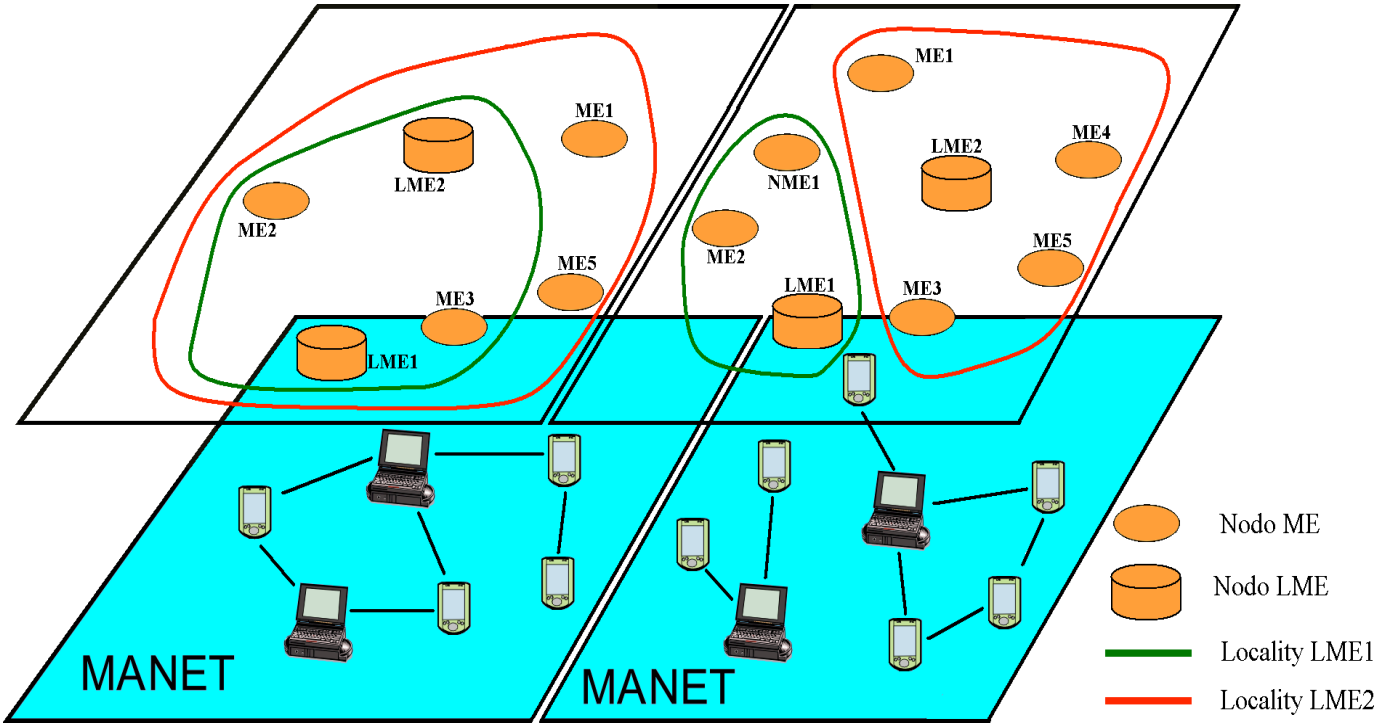


Figura 2.4 Località parzialmente sovrapposte.

La Figura 2.4 mostra una situazione operativa dove LME1 e LME2 appartengono allo stesso gruppo e definiscono due località che sono

parzialmente sovrapposte. I membri che si trovano nell'intersezione delle due località sono inclusi sia nella vista di LME1 che in quella di LME2.
JOINING. Consideriamo il caso di una entità non membro NME1 che entra nell'intersezione tra la località di LME1 e la località diLME2; NME1 può usare il PENS per ottenere la lista di tutti gli LME presenti nella località. In questo caso la lista conterrà LME1 e LME2 insieme con i loro GIDs, PIDs e IPs. Il J/LMS di NME1 riconosce che LME1 e LME2 appartengono allo stesso gruppo e può decidere di inviare la richiesta di join a uno di questi due (supponiamo LME1). Il J/LMS di LME1 riceve la richiesta di join (che contiene i profili utente e del dispositivo di NME1) e confronta se le preferenze di gruppo espresse da NME1 matchano con il profilo di gruppo. Se il confronta da esito positivo LME1 invia un messaggio di ack che contiene gli identificatori GID/PID a NME1 e include NME1 nel gruppo (inoltre si coordina con il VMS per aggiungere NME1 nella vista del gruppo). In questo momento NME1 è incluso solo nella vista di LME1 ma trovandosi nell'intersezione delle due località riceverà sia le viste di LME1, che quelle di LME2. Quando riceve la vista gestita da LME2 riconosce che non vi è incluso, così gli invia le sue informazioni di profilo (ottenute dal J/LMS). Dopo averne ricevuto il profilo LME2 include NME1 nella sua vista.
PROPAGAZIONE DELLE VISTE. LME1 e LME2 non appartengono alla stessa località e di conseguenza non si comunicano le loro viste reciprocamente. Il VCS sia di LME1 che di LME2 permette ai rispettivi VMS di propagare le viste ai membri dalle loro località. Tutti i membri che si trovano nell'intersezione tra la località di LME1 e la località diLME2 ricevono sia la vista di LME1 che quella di LME2 ed effettuano una fusione di queste sulla base degli identificatori GID/PID.
ROAMING. Consideriamo il caso in cui un membro ME3 appartenente allo stesso gruppo di LME1 e di LME2 entra nell'intersezione tra la località di LME1 e la località di LME2. L'istanza del VMS di ME3 riceve le viste aggiornate inviate da LME1 e LME2 e riconosce che non è incluso in nessuna delle due. Al fine di essere incluso in entrambe le viste ME3 invia le sue informazioni di profilo sia a LME1 che a LME2. Dopo averne ricevuto il profilo i due LMEs includono ME3 nelle loro viste.

2.3.3 **Località distinte**



(a) (b)

Figura 2.5 Località distinte.

La figura 2.5 (a) mostra una situazione operativa dove due LMEs (LME1 e LME2) appartengono allo stesso gruppo e definiscono due località innestate, con la località di LME1 contenuta nella località di LME2.
A causa della mobilità dei dispositivi la località di LME1 si separa da quella di LME2 figura 2.5 (b).
JOINING. Consideriamo il caso di una entità non membro NME1 che entra nella località definita da LME1 e si appresta a fare il join ad un gruppo di suo interesse. L'istanza J/LMS del dispositivo NME1 ottiene la lista di tutti gli LMEs presenti nella sua località, cioè solo LME1. Il J/LMS di NME1 invia la richiesta di join al gruppo a LME1. Simili operazioni di join avvengono se la entità non membro si trova nella località definita da LME2.
PROPAGAZIONE DELLE VISTE. Come mostrato in figura 2.5 (b), LME1 e LME2 non sono co-locati e formano due località distinte. Ne deriva che LME1 e LME2 non possono comunicarsi reciprocamente le loro viste. Il VCS di entrambe i dispositivi permette al rispettivo VMS di propagare la vista all'interno della propria località.
ROAMING. Consideriamo ora una entità membro ME1 appartenente allo stesso gruppo di LME1 e LME2 che si muove dalla località di LME1 alla località di LME2. Da una parte, il PENS di LME1 smette di ricevere i beacon trasmessi dal PS di ME1 e, di conseguenza, il VMS di LME1

rimuove la entry associata a ME1 dalla vista. Dall'altra parte, l'istanza VMS di ME1 riceve la vista aggiornata trasmessa da LME2 e riconosce che non vi è incluso. Per aggregarsi alla località di LME2, ME1 invia le sue informazioni di profilo a LME2. Dopo averne ricevuto il profilo, LME2 include ME1 nella sua vista.

CAPITOLO 3

Il View Manager Service

In uno scenario MANET, dove la struttura della rete è estremamente mutevole, il problema dell'interazione e della coordinazione dei

componenti è particolarmente significativo. Nuovi nodi si possono aggiungere o sottrarre dalla rete e la topologia della rete stessa può cambiare in modo imprevedibile. Per questo motivo i componenti devono essere stabiliti dinamicamente in fase di esecuzione: tutte le entità devono poter beneficiare della visibilità dell'ambiente circostante (context-awareness) al fine di interagire e coordinare le attività in modo flessibile e robusto.
AGAPE riconosce due tipi di componenti in base alle caratteristiche e al ruolo che possono assumere: Managed Entity (ME) e Locality Manager Entity (LME). Gli ME sono i componenti caratterizzati da scarse risorse computazionali, che sfruttano i servizi di gestione di gruppo forniti da AGAPE per collaborare con le altre entità. Gli LME, oltre a collaborare con le altre entità, supportano le operazioni di gestione dei gruppi in favore degli LME: favoriscono la creazione run-time di un nuovo gruppo, permettono alle entità di unirsi al gruppo e mantengono una lista aggiornata dei membri del gruppo co-locati (context-dependent view).

3.1 Concetto di vista

Il meccanismo di coordinazione proposto da AGAPE per offrire ai componenti informazioni sull'ambiente circostante si basa sul concetto di vista. Una vista è un insieme di elementi appartenenti ad uno stesso gruppo che si trovano in una certa località, quella dell'LME che la gestisce.
In particolare, ogni vista include gli identificatori GID/PID e il livello batteria dell'LME che l'ha generata ed è composta da una lista di elementi. Per ogni elemento appartenente alla vista vengono specificati l'identificativo personale (il PID ottenuto dal PENS) e le informazioni di profilo che si riferiscono all'utente o el dispositivo (ottenute dal JLMS).
L'LME, gestore della vista, permette agli elementi presenti nella sua località di unirsi o sganciarsi da un gruppo, aggiornando di conseguenza la corrispondente vista; tale vista viene disseminata periodicamente in broadcast a tutte le entità, in modo che queste possano prendere coscienza in tempo reale degli elementi presenti nell'ambiente circostante con i quali hanno la possibilità di interagire. Inoltre, nel gestire una vista, un LME deve garantire la consistenza di questa in caso di roaming: deve riconoscere (e quindi aggiungere nella sua vista) un elemento che ha il GID di un gruppo gestito e che in un determinato momento si sposta nella

sua località. In particolare, ogni elemnto che riceve una vista di un suo gruppo controlla di essere incluso nella vista; se non è presente invia il proprio profilo all'LME che ha disseminato la vista affinchè vi venga inserito.
Il componente di AGAPE che offre i servizi di gestione della vista è il VMS (View Manager Service).

3.2 Requisiti e casi d'uso

Come mostrato in figura 3.1, le funzionalità che il VMS deve supportare, si defferenziano a seconda del tipo di entità: in particolare possiamo notare che un LME estende le funzionalità di un ME.

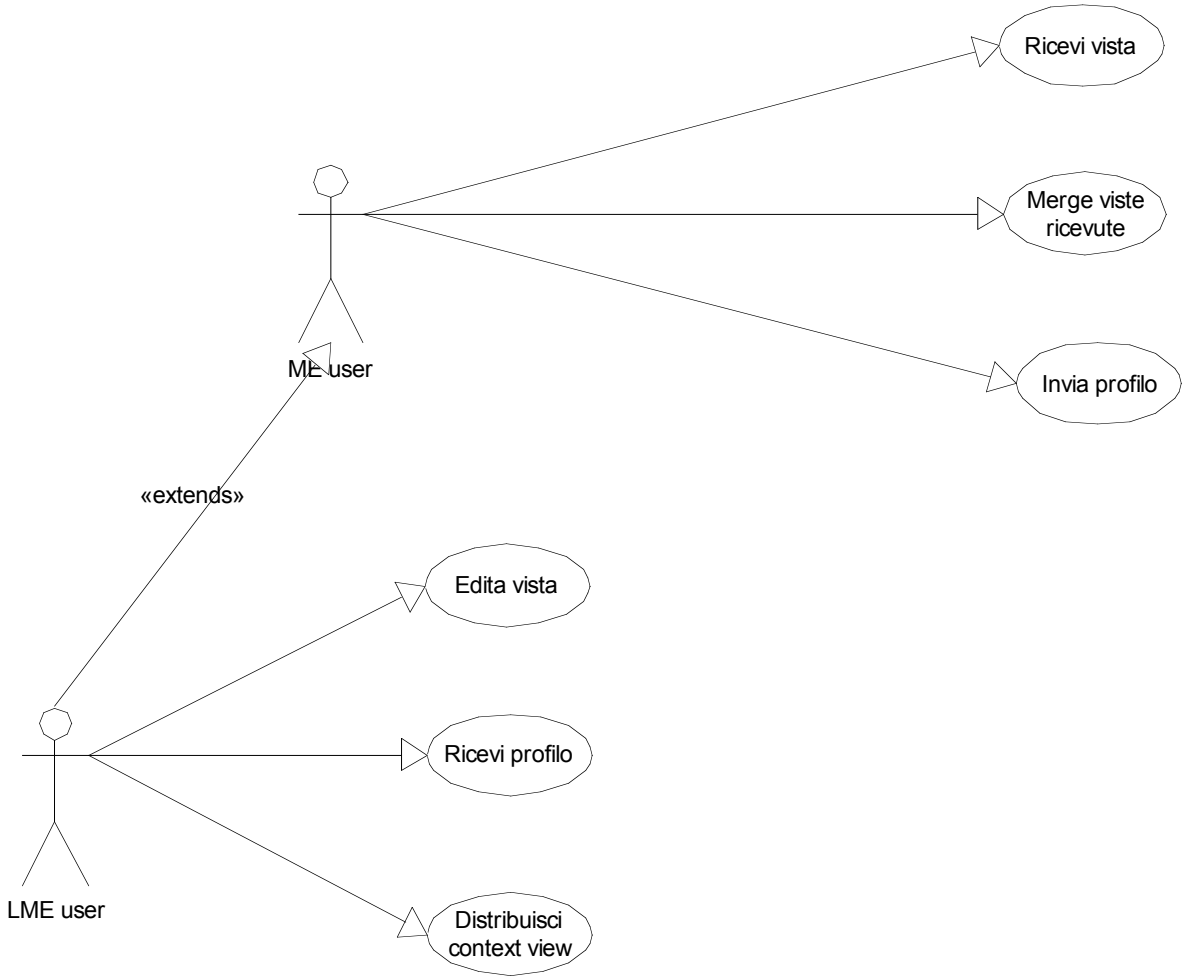


Figura 3.1 Casi d'uso del View Manager Service

Il VMS di un ME si occupa fondamentalmente di ricevere le viste del proprio gruppo, che sono trasmesse dagli LME presenti nella sua località e

di fonderle per ottenere una visione completa dell'ambiente circostante. Quando l'ME riceve una vista di un suo gruppo in cui non è presente invia il proprio profilo all'LME che ha disseminato la vista. Alla ricezione del profilo l'LME aggiungerà nella sua vista l'ME mittente.
I servizi veri e propri di gestione della vista sono forniti nel VMS degli LME. Infatti è l'LME che permette la creazione di un gruppo, e quindi di una vista, e favorisce l'inserimento o la rimozione di un membro della vista in seguito ad una richiesta di join/leave. Oltre a tenere aggiornata la vista, il VMS di un LME la dissemina periodicamente nella sua località.
Sono state implementate due versioni del VMS: una completa che supporta tutti i servizi di gestione delle viste, destinata agli LME, e una limitata che fornisce i servizi di base propri degli ME.

3.3 **LME (Locality Manager Entity)**

Un LME fornisce le funzionalità per creare un gruppo, consentire ad un elemento di entrare a far parte di un gruppo (join) o di uscire da un gruppo. In particolare, in questi casi, il VMS di un LME consente di inizializzare una vista, di aggiungere elementi alla vista o di rimuoverli.
L'LME gestisce tante viste quanti sono i gruppi a cui ha fatto il join o che ha inizializzato. Ogni vista viene periodicamente aggiornata in modo che contenga tutti i membri che si trovano nella località dell'LME e che appartengono allo stesso gruppo.
L'LME gestisce anche una lista di viste che riceve dagli altri LME; ovviamente si tratta di viste che si riferiscono ad un gruppo gestito e che sono state inviate da LMEs co-locati. In particolare, quando riceve una vista di un suo gruppo, un LME scandisce tutti gli elementi della vista ricevuta per aggiungere quelli che, pur trovandosi nella sua località, non sono presenti nella sua vista. Inoltre alla ricezione della vista di un proprio gruppo controlla se egli stesso è presente o meno nella lista. Se non è presente, invia il proprio profilo all'LME che ha disseminato la vista in modo che questo possa aggiungerlo.

3.3.1 **Inizializza vista**

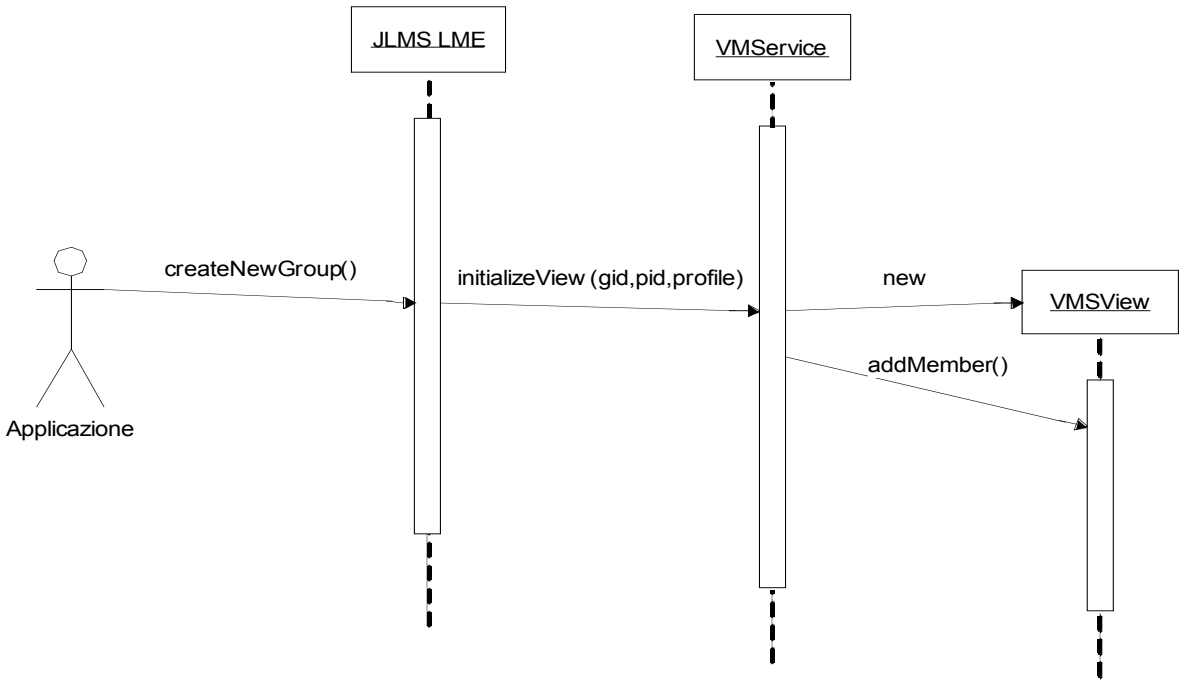


Figura 3.2 Inizializza vista

Un LME ha la possibilità di creare un nuovo gruppo. Per fare questo l'applicazione invoca il metodo *createNewGroup()* del JLMS. Dopo aver creato gli identificatori GID, PID sfruttando il PENS, il JLMS invoca il metodo *initializeView(gid, pid, profile)* del VMService. Quindi il VMS crea una nuova vista, inizializzandola con i suoi identificatori e il suo livello batteria, e aggiunge se stesso come primo elemento della vista. Da questo momento in poi gli elementi che si trovano nella sua località hanno la possibilità di fare il join al nuovo gruppo.

3.3.2 **Join ad un gruppo**

Quando un elemento vuole entrare a far parte di un gruppo invia, attraverso il JLMSJoinThread, una richiesta di join ad un LME che si trova nella sua località. Dopo aver ricevuto la richiesta, l'istanza JLMS dell'LME controlla se gli attributi di gruppo matchano con le preferenze espresse dall'utente. Se tale controllo dà esito positivo, il JLMS dell'LME permette all'elemento di entrare nel gruppo e gli invia un messaggio di Ack che include gli identificatori GID/PID, il profilo completo del gruppo e la vista del gruppo che recupera dal VMS.

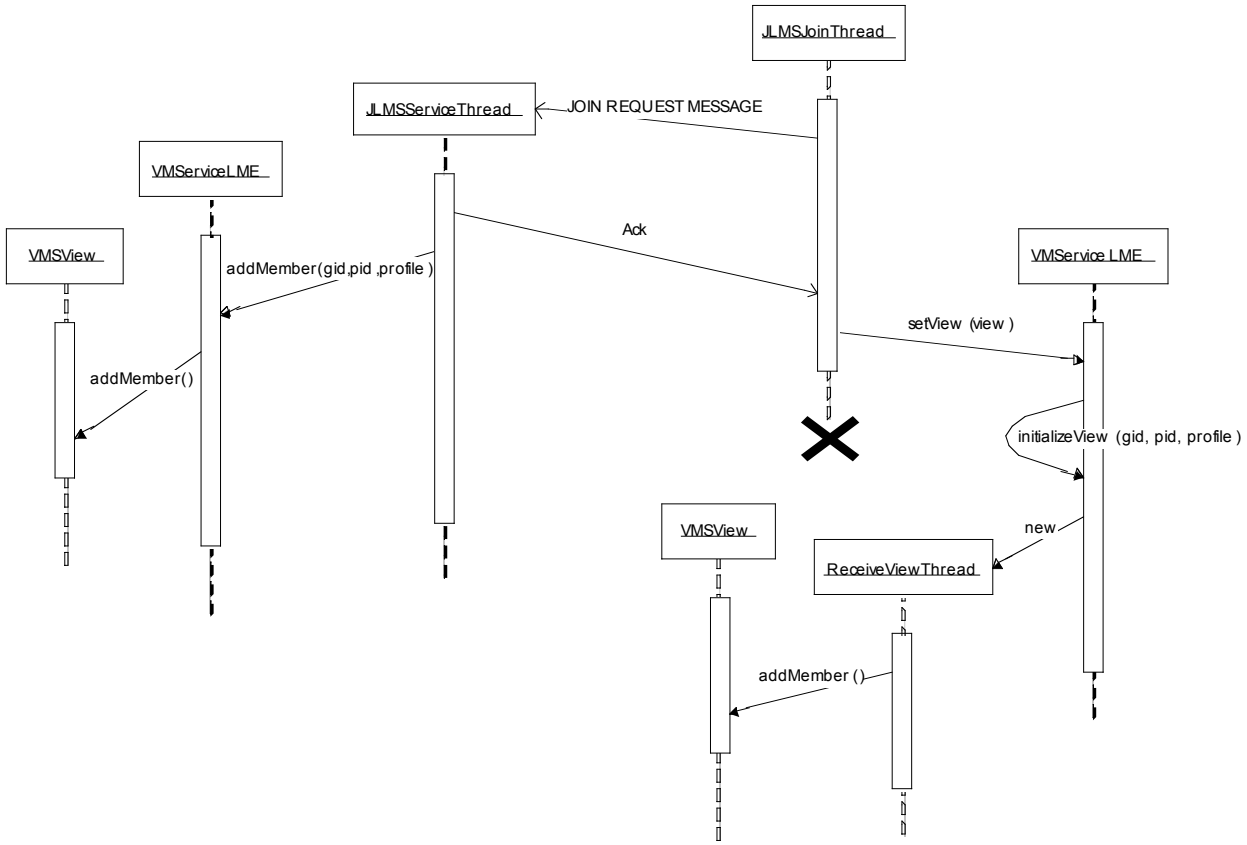


Figura 3.3 Join ad un gruppo

In più il JLMS dell'LME si coordina con il VMS per includere il profilo del nuovo membro nella vista, invocando il metodo *addMember(gid, pid, profile)*, che aggiorna la vista relativa a quel gruppo.
L'elemento che riceve l'Ack, sempre attraverso il thread JLMSJoinThread, invoca il metodo *setView(view)* del VMS.
In figura 3.3 è mostrato il comportamento di un LME che riceve l'ack. Il VMS inizializza una nuova vista con gli identificatori GID,PID e il profilo che ha ricevuto nell'Ack, aggiungendo in questo modo anche se stesso, e invoca il ReceiveViewThread che aggiunge gli elementi della vista contenuta nell'Ack alla vista appena creata. Da questo momento in poi l'LME ha una nuova vista da gestire e da propagare.

3.3.3 **Leave dal gruppo**

In tutte le entità, sia LME che ME, l'operazione di leave da un gruppo si effettua attraverso il metodo *leaveFromGroup(gid)* del JLMS; questo si coordina con il PS affinché smetta di propagare i propri identificatori GID/PID relativi a quel gruppo. Inoltre il JLMS si coordina con il VMS invocando il metodo *deleteViews(gid)*. In questo modo il VMS di un LME che ha effettuato il leave da un gruppo rimuove la corrispondente vista gestita e cancella tutte le viste ricevute relative a quel gruppo.

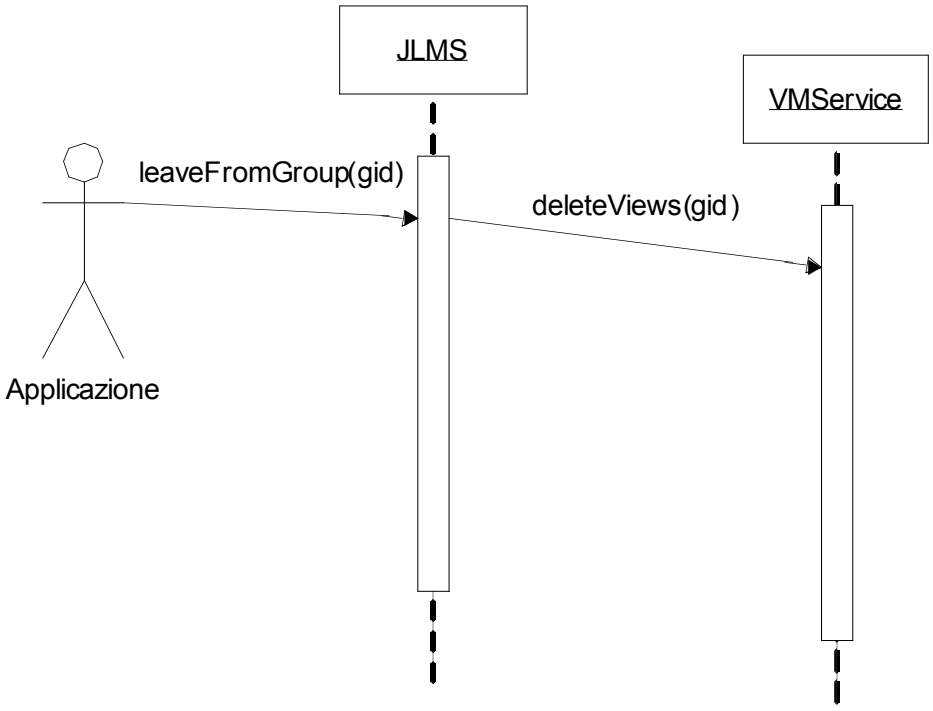


Figura 3.4 Leave da un gruppo (Caso LME)

3.3.4 **Dissemina vista**

La diffusione delle viste avviene periodicamente in broadcast, sfruttando le facility implementate dall'NMS, ed è effettuata da un LME che ha avuto il permesso dal VCS. Prima di disseminare la vista infatti il VMS si coordina con il VCS al fine di evitare propagazioni inutili.
Il VMS di un ME inserisce in una lista le viste ricevute che si riferiscono al suo gruppo attuale. Dopo un intervallo di tempo (pari a quello con cui l'LME dissemina le viste) effettua attraverso il MergeReceivedView il merge delle viste ricevute e setta la vista attuale.

Nell'LME un *ActiveComponent*, di nome ReceiveView, è costantemente in attesa di ricevere viste trasmesse in broadcast da LME co-locati; dopo aver ricevuto una vista ed aver constatato che si tratta di una vista di un gruppo gestito viene lanciato il MenageViewThread. Questo thread dopo aver memorizzato la vista ricevuta, invoca il ReceiveViewThread che si occupa di inserire nella propria vista i membri della vista ricevuta che si trovano nella sua località.

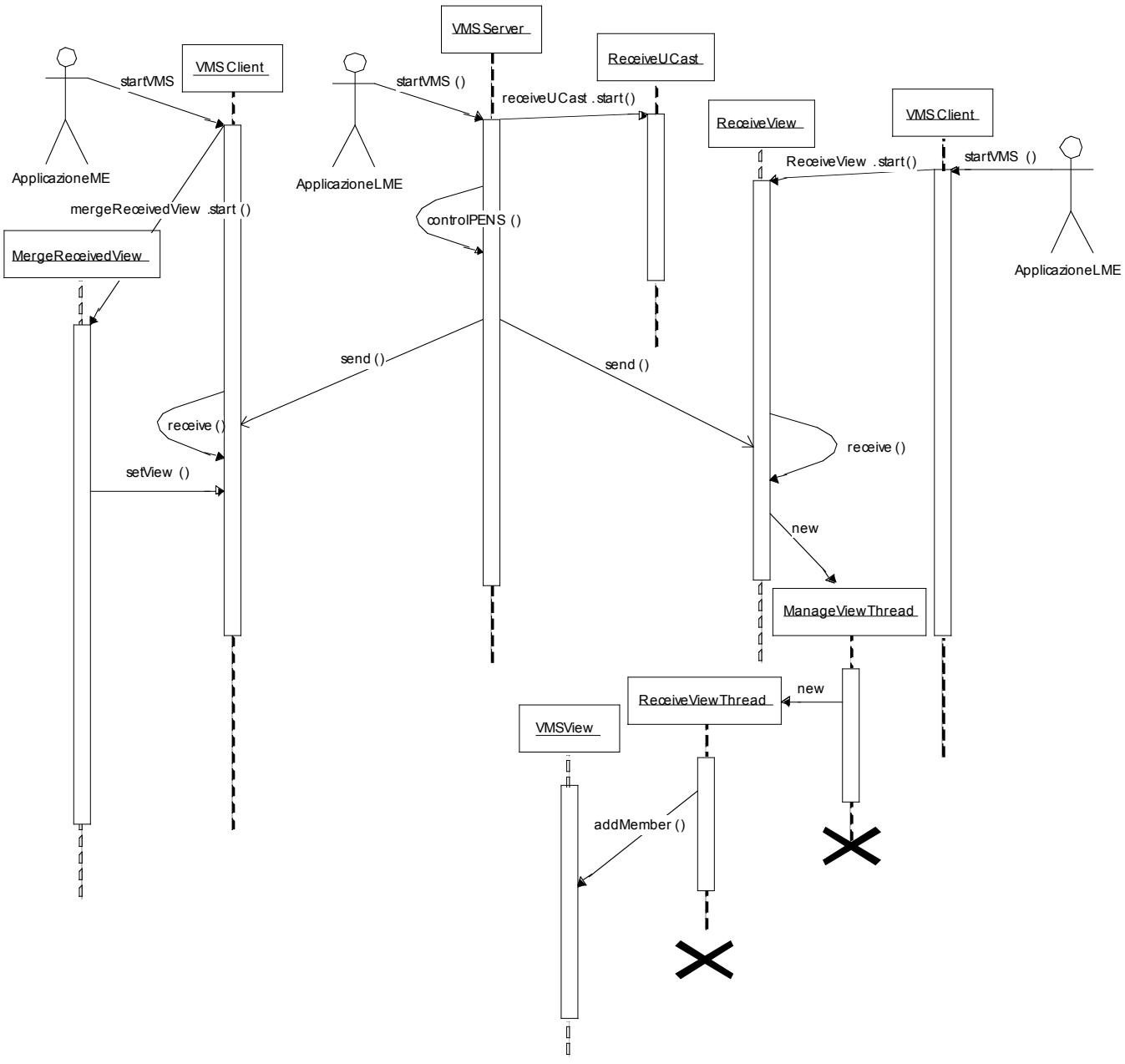


Figura 3.5 Dissemina vista

3.3.5 **Roaming**

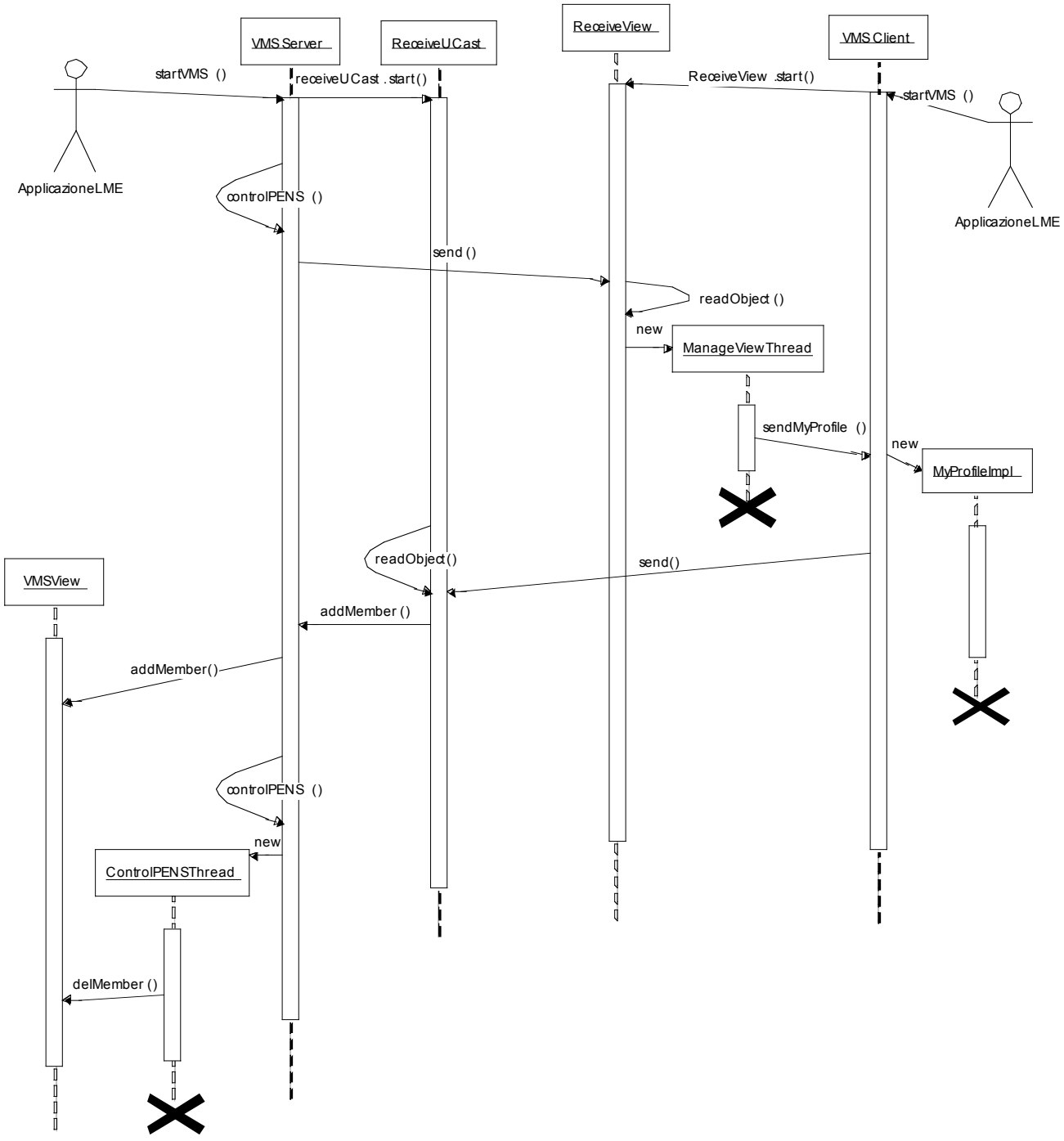


Figura 3.6 Roaming

Come già detto, periodicamente il VMService di un LME controlla il PENS e invia in broadcast le viste.
Se un elemento, sia LME che ME, riceve una vista che si riferisce al suo gruppo e si accorge di non essere presente nella vista, invia in unicast il

proprio profilo all'LME che ha disseminato tale vista. In questo modo l'LME che riceve il profilo inserisce nella vista il nuovo elemento.
Se, controllando il PENS, l'LME scopre che un elemento della sua vista non è più presente nella sua località, lancia il ControlPENSThread che dopo qualche secondo (10), interroga nuovamente il PENS ed eventualmente rimuove l'elemento dalla vista.

3.4 **ME (Menaged Entity)**

Un ME, caratterizzato da limitate capacità di memoria e di calcolo, non dispone delle funzionalità complete del VMS, né del JLMS e né del VCS; non è in grado di creare un gruppo, né si occupa di tenere aggiornata la vista della sua località o di distribuirla ai suoi vicini.
L'ME riesce ad avere un visione dell'ambiente circostante per mezzo delle viste ricevute dagli LME co-locati: unisce le viste ricevute in certo intervallo di tempo e che si riferiscono al suo gruppo attuale per settare la sua vista. Inoltre, come succede per gli LME, alla ricezione della vista di un proprio gruppo controlla se egli stesso è presente o meno nella lista. Se non è presente, invia il proprio profilo all'LME che ha disseminato la vista in modo che questo possa aggiungerlo.
Un ME, anche se può unirsi a più gruppi, ha un solo GID attuale e una sola vista, quella che si riferisce all'ultimo gruppo a cui ha fatto il join, o quella di un gruppo a cui fa lo switch (previsto anche nel caso del leave dal gruppo attuale).

3.4.1 **Join ad un gruppo**

In figura 3.7 è mostrato il comportamento di un ME che riceve l'ack in seguito ad una richiesta di join ad un gruppo.
Il VMS cambia il valore del GID attuale e azzera la vista attuale; in questo modo verranno ricevute le viste del nuovo gruppo e la vista attuale verrà settata di conseguenza.

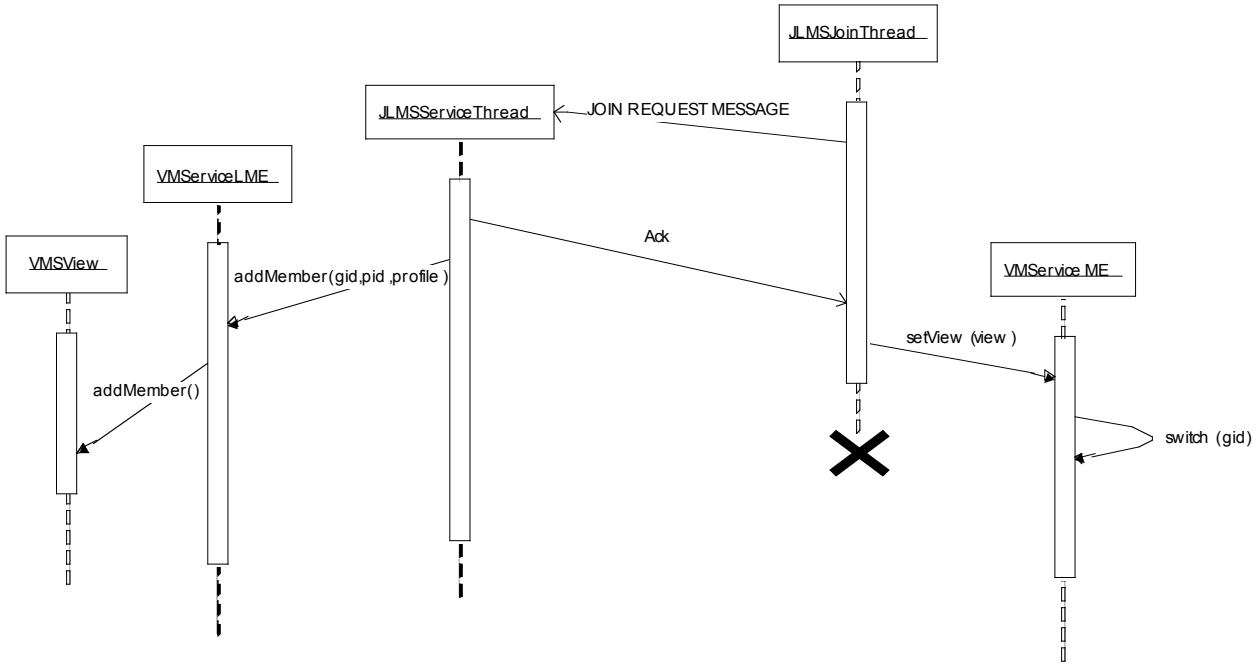


Figura 3.7 Join ad un gruppo

3.4.2 **Leave dal gruppo**

Quando un ME fa il leave dal gruppo attuale ha la possibilità di inserire un gid che rappresenta il gruppo a cui fa lo switch. Dal momento in cui effettua il leave, l'ME ignora le viste relative a quel gruppo che riceve dagli LME che si trovano nella sua località.

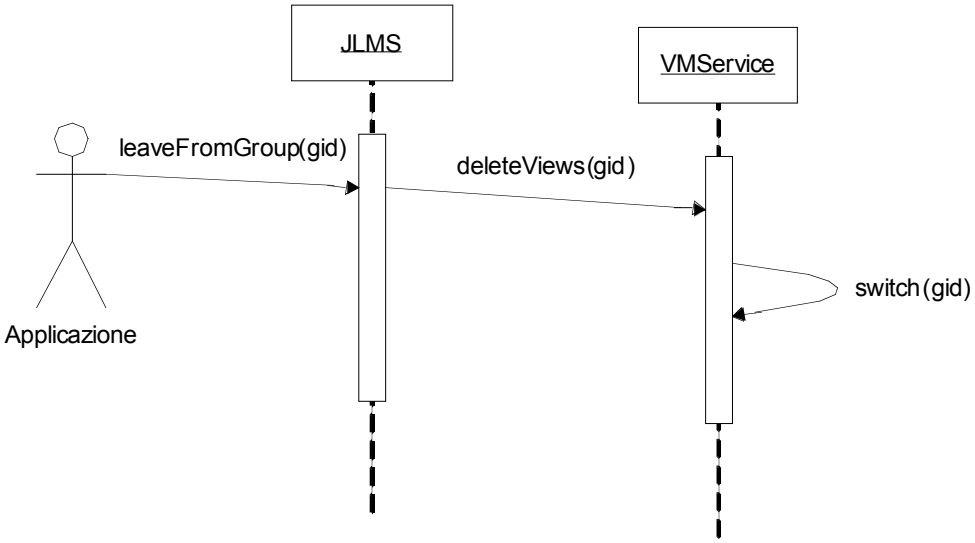


Figura 3.8 Leave da un gruppo (Caso ME)

3.4.3 **Merge viste ricevute**

Quando è attivo, un ME legge sulla porta di input broadcast in attesa di ricevere eventuali viste che può LME che si trovano nella sua località. Memorizza le viste ricevute in un certo intervallo di tempo e che si riferiscono al suo gruppo attuale in un vettore. Periodicamente, un componente attivo, il MergeReceivedView setta la vista attuale dell'ME in base alle viste ricevute e azzera il vettore. In questo modo la vista attuale dell'ME viene creata dinamicamente effettuando una fusione delle viste ricevute nell'ultimo intervallo di tempo. L'ME potrà utilizzare la vista per capire quali sono gli elementi con cui può interagire conoscendone i rispettivi profili.

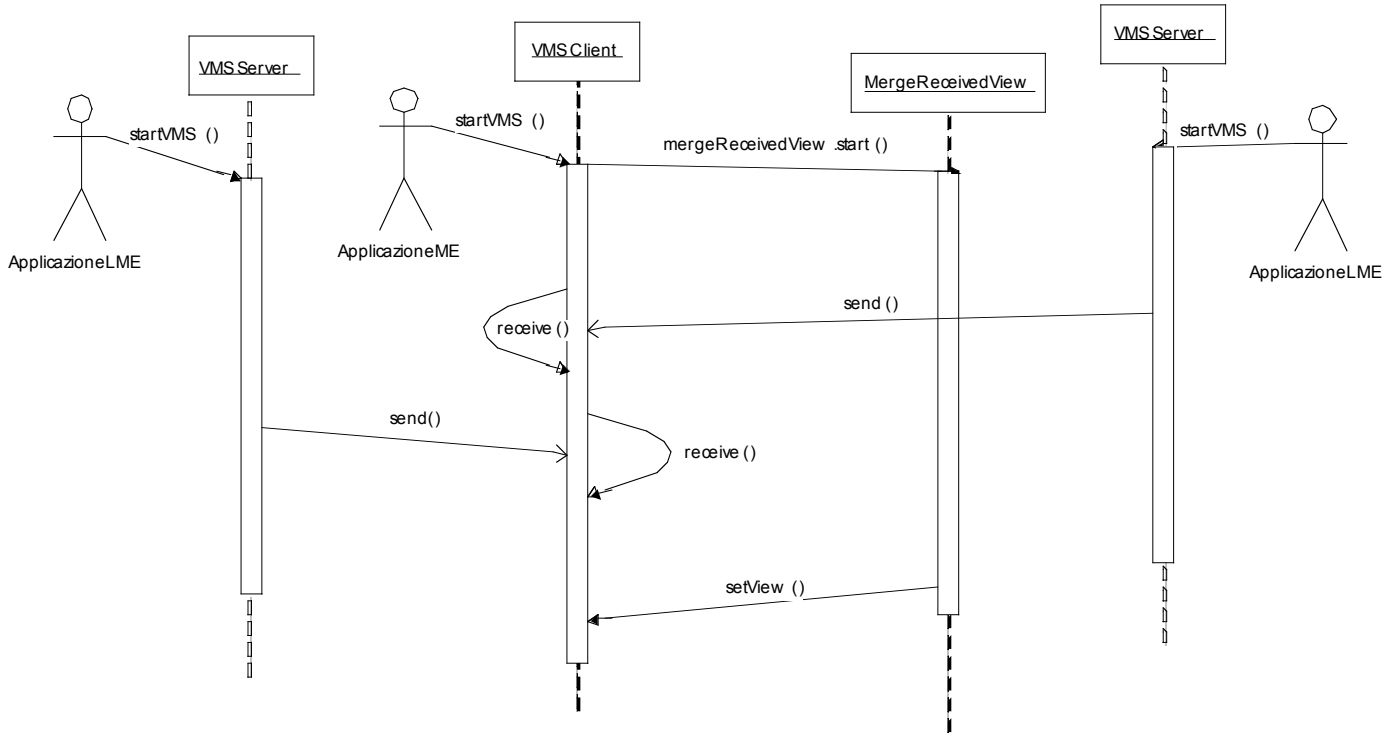


Figura 3.9 Merge viste ricevute

3.5 – **Implementazione del VMS LME**

Affinché gli elementi co-locati abbiano visione dell'ambiente circostante il VMS dell'LME periodicamente invia le viste gestite agli elementi presenti nella propria località. L'invio è effettuato in broadcast sfruttando le facility implementate dall'NMS. Ovviamente anche un LME riceve le viste trasmesse in broadcast dagli altri LME situati nella sua località e memorizza quelle che si riferiscono ai gruppi gestiti; per far questo utilizza un apposito ActiveComponent: il ReceiveView. Inoltre se non è presente in una vista ricevuta che si riferisce ad un gruppo gestito, il VMS invia in unicast il proprio profilo all'LME che ha disseminato la vista affinché vi venga inserito.
Inoltre, il VMS dell'LME si coordina con il PENS per tenere aggiornate le viste: controlla che tutti gli elementi delle sue viste siano presenti nel PENS, altrimenti li rimuove, e controlla che tutti gli LME gestori delle viste ricevute siano visibili nel PENS, altrimenti rimuove la corrispondente vista ricevuta dalla lista,
Utilizzando un altro ActiveComponent, il ReceiveUCast, può ricevere in unicast i profili di elementi da aggiungere a una vista gestita. Sono profili di elementi, LME o ME, che appartengono allo stesso gruppo e trovandosi nella stessa località hanno precedentemente ricevuto la vista nella quale non erano presenti.

3.5.1 **Diagramma delle classi LME**

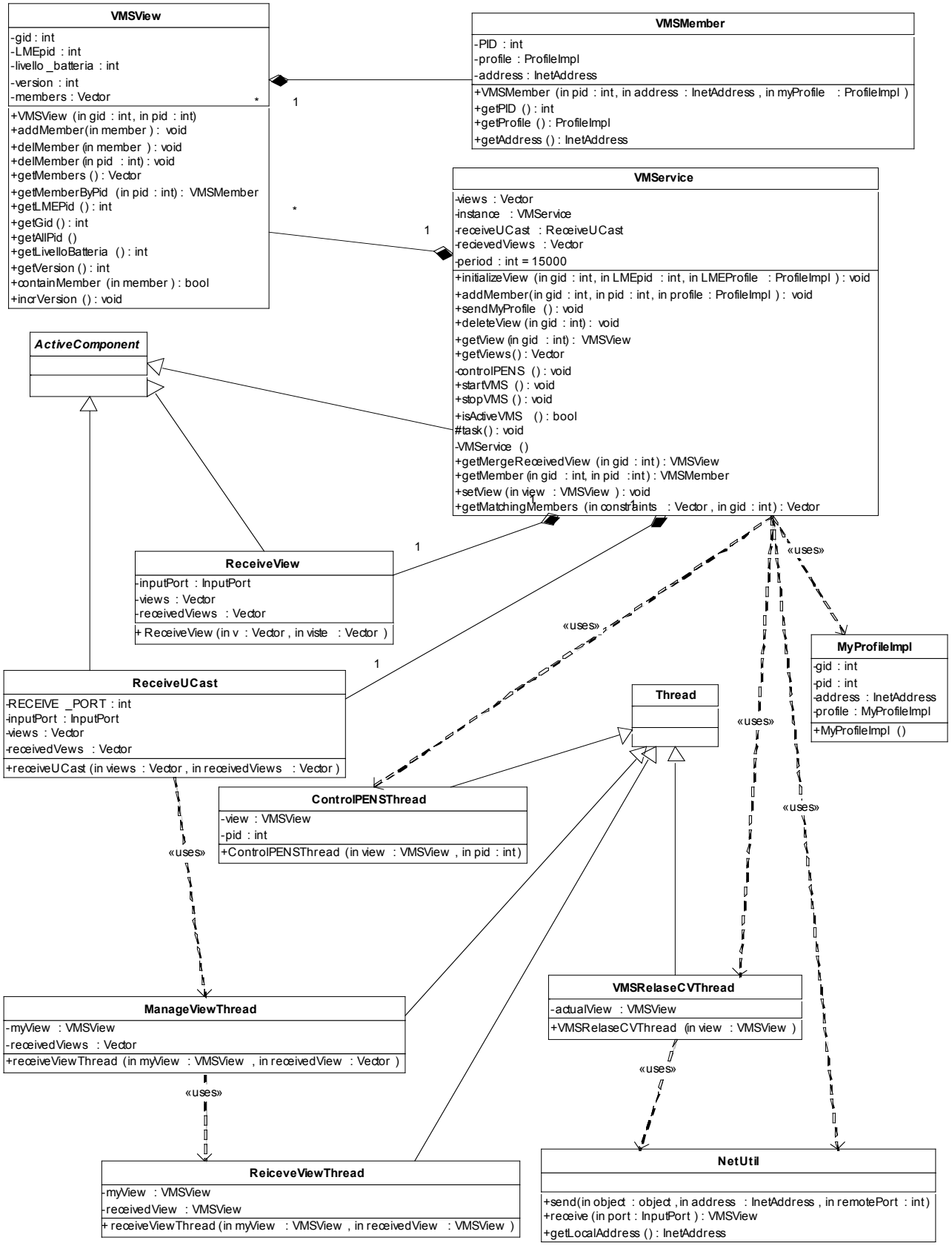


Figura 3.10 Diagramma delle classi LME

3.5.2 **Responsabilità delle classi**

La classe VMService realizza il VMS, ovvero il componente attivo addetto alla gestione e alla trasmissione periodica delle viste. Ad ogni trasmissione la versione della vista viene incrementata.
Una vista è realizzata dalla classe **VMSView**, che oltre a contenere l'insieme di membri appartenenti ad uno stesso gruppo e co-locati all'LME gestore, include gli identificatori GID/PID e il livello batteria del dispositivo che l'ha generata. All'interno di una vista, ma anche all'interno di un gruppo, i membri hanno il PID univoco.
VMSMember è la classe che rappresenta un membro della vista. E' caratterizzato da un PID, un profilo e un indirizzo.
La classe **VMService** è l'ActiveComponent che fornisce il servizio di gestione delle viste. Può gestire più viste, una per ogni gruppo a cui ha fatto il join , e inizializzare viste di nuovi gruppi.
Tiene traccia anche delle viste ricevute, ovviamente solo quelle relative a gruppi che gestisce, al fine di coordinarsi con il VCS per evitare trasmissioni di viste inutili.
Si occupa di:

- Distribuire le context view in broadcast a intervalli regolari (ogni 15 secondi)
- Tenere aggiornate le context view controllando periodicamente il PENS: elimina i membri delle viste per i quali non vede l'entry nel PENS,
- Tenere aggiornato l'elenco delle viste ricevute eliminando quelle per le quali non vede l'LME gestore nel PENS.

La classe **ReceiveUCast** rappresenta l'ActiveComponent che legge continuamente sulla porta unicast al fine di ricevere i profili inviati da elementi che devono essere inseriti in una vista gestita dall'LME.
La classe **ReceiveView** è l'ActiveComponent che legge continuamente sulla porta broadcast al fine di ricevere le viste inviate dagli LME che si trovano nella sua località. Quando viene ricevuta la vista di un gruppo gestito viene fatto partire il ManageViewThread.
La classe **ManageViewThread** è il thread che parte quando viene ricevuta una vista di un gruppo gestito. Si occupa di

- Tenere aggiornato l'elenco delle viste ricevute,
- Inviare il proprio profilo all'LME che ha disseminato la vista qualora non fosse presente nella vista ricevuta.

- Fa partire il ReceiveViewThread

La classe **ReceiveViewThread** è il thread che si occupa di inserire nella vista gestita i membri contenuti nella vista ricevuta che appartengono anche alla propria località. Viene invocato dal ManageViewThread quando ricevo la vista di un mio gruppo o dal metodo VMService.setView(view) quando l'LME fa il join ad un gruppo. Infatti, dopo aver ricevuto l'ack per il join, l'LME inizializza una vista con il GID di quel gruppo e, dopo aver aggiunto se stesso, aggiunge i membri della vista contenuta nell'ack usando il ReciveViewThread (vengono aggiunti solo quelli che sono nella sua località) (vedi figura 3.3).
La classe **VMSRelaseCVThread** è il thread utilizzato per disseminare una vista. Scandisce tutti gli elementi della vista, ne recupera l'indirizzo e utilizzando la primitiva NetUtil.send() invia in unicast la vista.
La classe **ControlPensThread** è il thread che ha il compito di togliere dalla vista un elemento che non è presente nel PENS. Viene invocato dal VMService. Prima di togliere il membro dalla vista però aspetta qualche secondo e interroga nuovamente il PENS. Questo consente di aspettare l'aggiornamento del PENS quando inserisco un membro nella vista in seguito ad un join. Infatti quando un LME riceve una richiesta di join, inserisce subito il membro nella vista e invia l'ack; l'elemento inserito inizierà a trasmettere i ps-beacon solo dopo aver ricevuto l'ack, e il PENS dell'LME si aggiorna con ritardo. Per evitare di togliere subito l'elemento appena inserito quindi il ControlPens aspetta un eventuale aggiornamento del PENS.
MyProfileImpl è la classe che racchiude il profilo, il GID, il PID, e l'indirizzo di un membro. Viene inviato ad un LME dal VMService di un elemento che riceve il VMSMessageProfileRequest. L'LME che riceve il MyProfileImpl inserirà nella vista corrispondente il nuovo membro.
NetUtil è la classe che contiene la primitiva statica *send (Object object, InetAddress address, int remotePort)* per inviare un qualsiasi oggetto in unicast all'indirizzo address sulla porta remotePort.

3.6 – **Implementazione del VMS ME**

Il VMS di un ME gestisce una lista di viste in cui inserisce quelle ricevute (che riguardano il gruppo attuale) in un intervallo di tempo pari al periodo con il quale l'LME invia le viste.
Periodicamente effettua una fusione (merge) delle viste ricevute per settare la vista attuale ed elimina le viste ricevute. Ovviamente se durante tale periodo di tempo non sono state ricevute viste del gruppo attuale, la vista attuale non avrà nessun membro.
Continuamente può riceve sulla porta di broadcast una vista trasmessa da un LME che si trova nella sua località. Se la vista ricevuta si riferisce al gruppo attuale verrà aggiunta ad un vettore di viste ricevute. Inoltre invia il proprio profilo all'LME che ha disseminato una vista relativa al gruppo attuale qualora non fosse incluso in tale vista.
L'operazione di switch ad un gruppo a cui era stato fatto precedentemente il join setta il nuovo gruppo attuale, in modo che la vista attuale sia settata in base alle viste ricevute che si riferiscono a quel gruppo.

3.6.1 **Diagramma delle classi ME**

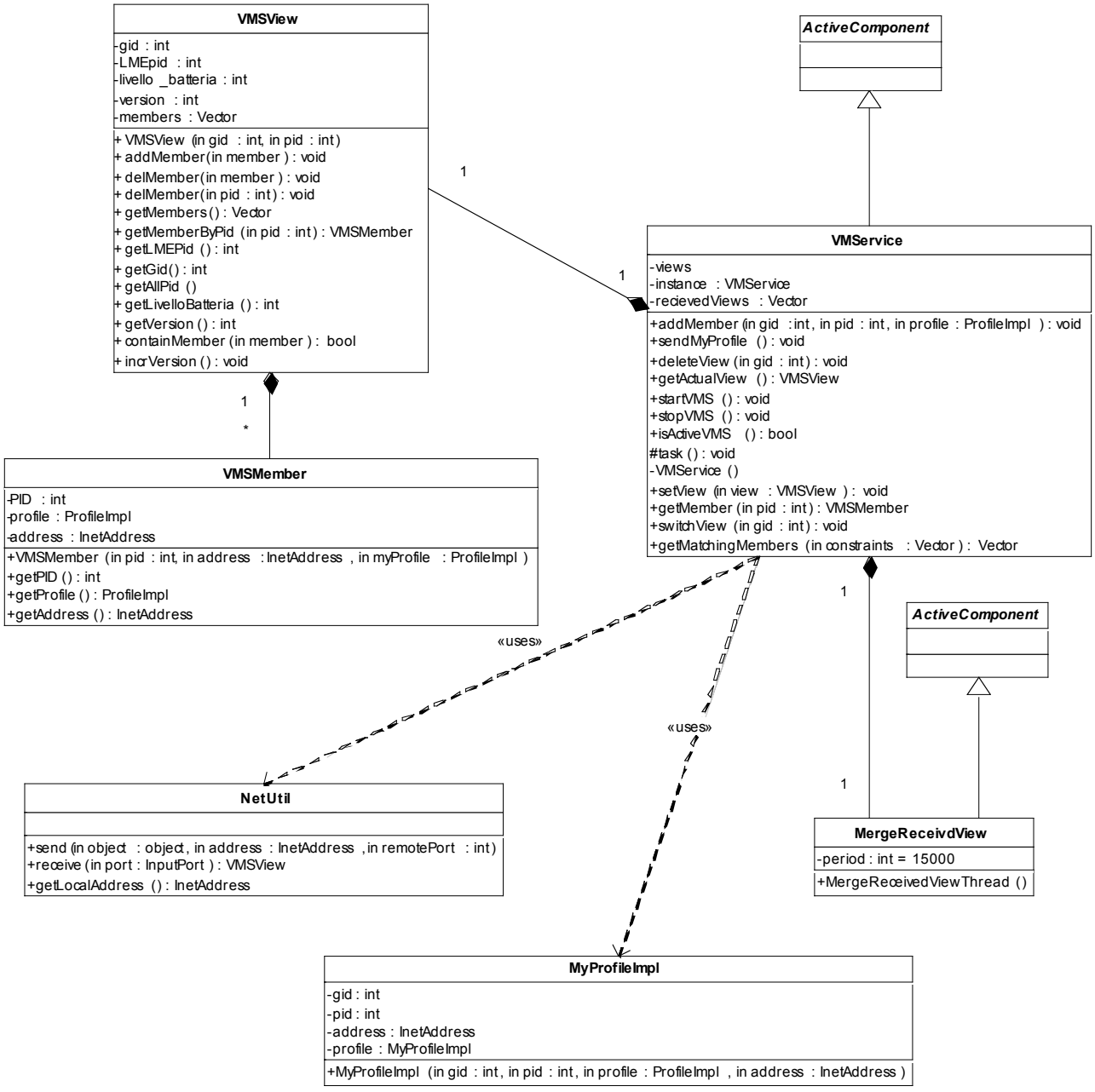


Figura 3.11 Diagramma delle classi ME

3.6.2 **Responsabilità delle classi**

Le classi VMSView, VMSMember, MyProfileImpl e NetUtil sono le stesse viste nell'LME e hanno le stesse funzionalità.
La classe **VMService** è l'ActiveComponent che gestisce la vista attuale.

Può gestire una sola vista, che si riferisce all'ultimo gruppo a cui ha fatto il join o lo switch.
Tiene traccia delle viste ricevute in un time-slot (in un intervallo di tempo pari al periodo con cui gli LME disseminano le viste), ovviamente solo quelle che si riferiscono al gruppo attuale. Queste viste saranno usate per settare la vista attuale.
Si occupa di ricevere le context view in broadcast ed aggiungere alla lista delle viste ricevute quelle che si riferiscono al gruppo attuale.
La classe **MergeReceivedView** è l'ActiveComponent che effettua il merge delle viste ricevute. Periodicamente, si occupa di:

- Fare il merge delle viste ricevute per settare la vista attuale.
- Eliminare le viste ricevute.

CAPITOLO 4

ESEMPIO APPLICATIVO

Al fine di testare il funzionamento di AGAPE abbiamo implementato un prototipo di applicazione di chat. Questa applicazione consente la comunicazione attraverso semplice scambio di messaggi tra utenti che appartengono allo stesso gruppo e che condividono gli stessi interessi. Ogni gruppo è caratterizzato da un profilo che ne determina obiettivi e caratteristiche. Gli utenti possono individuare i gruppi presenti in un determinato momento e hanno la possibilità di creare gruppi nuovi, o di entrare a far parte di un gruppo esistente o di uscirne. Ogni utente del sistema è caratterizzato dal proprio profilo, che ne modella attributi e preferenze. In base alle caratteristiche del profilo e ai vincoli associati al gruppo durante la sua creazione, l'applicazione concede o nega il permesso ad un utente che ha inviato una richiesta per entrare a far parte di un gruppo.
L'insieme di utenti con i quali è possibile comunicare viene mantenuto costantemente aggiornato dall'applicazione stessa e l'utente può usufruirne per selezionare i destinatari dei propri messaggi: comandi distinti permettono di inviare messaggi a tutti gli elementi del gruppo, ad uno solo scelto a livello applicativo o a tutti quelli i cui profili matchano con determinati vincoli espressi dall'utente. L'applicazione può quindi sfruttare l'elenco degli elementi che appartengono al gruppo e che in quel determinato momento sono "disponibili" per scegliere utenti con cui interagire.

4.1 **Principali comandi e funzionamento**

L'insieme di comandi che realizza la nostra semplice applicazione comprende:

- un comando **create** per creare un gruppo e inizializzare la corrispondente vista,
- un comando **join** per entrare a far parte di un gruppo,
- un comando **leave** per uscire da un gruppo,
- un comando **get group all** per avere la lista dei gruppi disponibili nella propria località,
- un comando **print views** che permette ad un elemento di stampare la lista che comprende gli elementi di tutti i gruppi a cui è associato(o soltanto una specificandone il GID),

- dei comandi per inviare messaggi ad un elemento della vista (**send)**, a tutti gli elementi di una vista (**send view**) o soltanto a quelli che nel proprio profilo hanno determinate caratteristiche (**send matching members**).

La nostra applicazione sfrutta le astrazioni messe a disposizione da AGAPE per realizzare questi comandi. Per prima cosa viene ripresa la gestione dei profili usata per descrivere le caratteristiche di utenti e dispositivi, ma anche dei gruppi che vengono creati. Le operazioni che permettono la creazione di un gruppo, l'ingresso o l'uscita di un elemento dal gruppo sono realizzate per mezzo delle funzionalità offerte dal JLMS. Per scoprire i gruppi presenti in una località un utente del sistema sfrutta il servizio di proximity messo a disposizione da AGAPE attraverso il PS e il PENS. La consistenza dell'insieme di utenti di un gruppo con i quali è possibile interagire è garantita dall'invio periodico di viste effettuato dal VMS. Per ogni elemento presente nella vista viene specificato il rispettivo profilo.
I messaggi sono inviati utilizzando le primitive di unicast dell'NMS e sono ricevuti da un ActiveComponent (il ReceiveMessage). Quando è attivo, il ReceiveMessage è sempre in attesa di messaggi sulla porta di ingresso unicast; quando un messaggio viene ricevuto viene stampato insieme agli identificativi del mittente.

4.2 **Implementazione**

Nei paragrafi successivi verrà descritto come i comandi sono stati implementati sfruttando le API di AGAPE. Particolare attenzione sarà dedicata alle API proprie del VMS.

4.2.1 **Creazione di un profilo**

Ogni utente è caratterizzato da un proprio profilo. Riportiamo un esempio di creazione di un profilo di utente che include gli attributi nome, cognome, età e professione.

```
ProfileDescriptionImpl profDesc = new ProfileDescriptionImpl();
ComponentDescriptionImpl compDesc = new ComponentDescriptionImpl();
compDesc.setLocalType("UTENTE");
```

```
AttributeDescriptionImpl attrDescName = new AttributeDescriptionImpl();
AttributeDescriptionImpl attrDescLastname=new AttributeDescriptionImpl();
AttributeDescriptionImpl attrDescAge = new AttributeDescriptionImpl();
AttributeDescriptionImpl attrDescProf = new AttributeDescriptionImpl();

attrDescName.setName("nome");
attrDescName.setComponentDescription(compDesc);
attrDescName.setResolution(1);
attrDescName.setBaseType(Class.forName("java.lang.String"));

attrDescAge.setName("age");
attrDescAge.setComponentDescription(compDesc);
attrDescAge.setResolution(2);
attrDescAge.setBaseType(Class.forName("java.lang.Integer"));

attrDescLastname.setName("cognome");
attrDescLastname.setComponentDescription(compDesc);
attrDescLastname.setResolution(1);
attrDescLastname.setBaseType(Class.forName("java.lang.String"));

attrDescProf.setName("professione");
attrDescProf.setComponentDescription(compDesc);
attrDescProf.setResolution(3);
attrDescProf.setBaseType(Class.forName("java.lang.String"));

compDesc.addAttributeDescription(attrDescName);
compDesc.addAttributeDescription(attrDescLastname);
compDesc.addAttributeDescription(attrDescAge);
compDesc.addAttributeDescription(attrDescPref);

profDesc.addComponentDescription(compDesc);

ProfileImpl profile = new ProfileImpl();
profile.setDescription(profDesc);

ComponentImpl component = new ComponentImpl();
component.setDescription(compDesc);
component.setName("informazioni utente");

LiteralAttributeImpl attrName = new LiteralAttributeImpl();
attrName.setDescription(attrDescName);
attrName.setComponent(component);
attrName.setValue("Mario");

LiteralAttributeImpl attrLastname = new LiteralAttributeImpl();
attrLastname.setDescription(attrDescLastname);
attrLastname.setComponent(component);
attrLastname.setValue("Rossi");

IntegerAttributeImpl attrAge = new IntegerAttributeImpl();
attrAge.setDescription(attrDescAge);
attrAge.setComponent(component);
attrAge.setValue("65");

LiteralAttributeImpl attrProf = new LiteralAttributeImpl();
attrProf.setDescription(attrDescProf);
attrProf.setComponent(component);
```

```
attrProf.setValue("pensionato");
component.addAttribute(attrName);
component.addAttribute(attrLastname);
component.addAttribute(attrAge);
component.addAttribute(attrProf);

profile.addComponent(component);
```

Codice d'esempio 4.1 – Creazione di un profilo

L'utente può modificare il valore di un attributo recuperando l'attributo dal profilo con il metodo *ProfileImpl.getAttribute(String name)* e settando il nuovo valore con il metodo *AttributeImpl.setValue(Object value)* (vedi Codice d'esempio 4.2).

```
static void incrAge()
{
   AttributeImpl  attr=(AttributeImpl)profile.getAttribute("age");
   Integer age = (Integer)attr.getValue();
   age = new Integer(age.intValue()+1);
   attr.setValue(age);
}
```

Codice d'esempio 4.2 – Modifica attributi di un profilo

4.2.2 **Attivazione dei servizi**

In primo luogo mostriamo i comandi invocati dalla nostra applicazione che servono per attivare i principali servizi di AGAPE: il PENS, il PS, il JLMS e il VMS. Inoltre viene attivato il componente che riceve i messaggi, il ReceiveMessage.

```
// start execution
PENService.startPENS();
PService.startPS();
JLMService.startJLMS();
VMService.startVMS();

ReceiveMessage.startRM();
```

Codice d'esempio 4.3 – Attivazione dei servizi

Ovviamente tali servizi vengono attivati all'inizio dell'esecuzione dell'applicazione.

4.2.3 **Creazione di un gruppo**

La creazione di un gruppo è effettuata dal JLMS attraverso il metodo statico *CreateNewGroup()* che ha come parametri di ingresso il profilo del gruppo (groupProfile) e un vettore v che rapresenta i vincoli che saranno utilizzati per discriminare gli elementi che tenteranno di associarsi al gruppo. Nell'esempio 4.4 i vincoli sono creati con il metodo statico *ConstraintFactory.createConstraint()*, ma l'utente ha anche la possibilità di settarli dinamicamente. Ogni gruppo ha un identificatore univoco, Gid, e un proprio profilo che ne caratterizza gli interessi e obiettivi.
Il metodo *CreateNewGroup*, coordinandosi con il PENS, genera l'identeficatore che verrà associato al gruppo e successivamente invoca il metodo statico *VMService.initializeView()* del VMS che inizializza una nuova vista con il Gid precedentemente creato; il VMS inserisce se stesso come primo elemento della vista e la gestisce fino a quando non effettua l'operazione di leave dal gruppo.

```
static void createNewGroup()
{
    System.out.println("Sono nel metodo create");
    ProfileFactory pf = new ProfileFactory();
    ProfileImpl groupProfile=pf.createTestGroupProfile();

    Vector v = ConstraintFactory.createConstraint();
    String s = JLMService.CreateNewGroup(groupProfile,v);
    System.out.println(s);
}
```

Codice d'esempio 4.4 – Creazione di un gruppo

4.2.4 **Associarsi ad un gruppo**

Per associarsi ad un gruppo un peer dell'applicazione invoca il metodo *JLMService.Join(gid)* del JLMS che provvede ad effettuare una richiesta di join al gruppo passato come argomento.

```
static void join(String joinString)
{
    int gid=Integer.parseInt(joinString);
    ProfileFactory pf=new ProfileFactory();
```

```
    ProfileImpl myProfile=pf.createTestProfile();
    JLMService.setMyProfile(myProfile);
    JLMService.Join(gid);
}
```

Codice d'esempio 4.5 – Associarsi ad un gruppo

4.2.5 **Disassociarsi da un gruppo**

Per disassociarsi da un gruppo un peer dell'applicazione invoca il metodo *JLMService.leaveFromGroup(gid)* del JLMS che si coordina con il PS affinché non trasmetta più i beacon corrispondenti.

```
static void leave(String leaveString)
{
    Integer gid=new Integer(leaveString);
    JLMService.leaveFromGroup(gid);
    System.out.println("Leave dal gruppo "+gid+"
                                          effettuato");
}
```

Codice d'esempio 4.6 – Disassociarsi da un gruppo

4.2.6 **Elencare i gruppi**

Per elencare i gruppi presenti nella propria località i peer utilizzano il metodo *printGroupsAll()* che sfrutta il PENS per ottenere la lista di tutti i gruppi presenti.

```
static void printGroupsAll()
{
    Enumeration enumGroups = PENService.getGroups();
    Integer current;
    while(enumGroups.hasMoreElements())
    {
        current = (Integer)enumGroups.nextElement();
        System.out.print("gid:"+current);
        System.out.println();
    }
}
```

Codice d'esempio 4.7 – Elencare gruppi presenti nella località

4.2.7 **Elencare gli elementi di una vista**

Dal momento in cui un elemento entra a far parte di un gruppo ha a disposizione una vista relativa a quel gruppo che contiene tutti gli elementi co-locati associati al gruppo. Per stampare tutti gli elementi della vista con i quali è possibile interagire si invoca il metodo *VMService.printView(int gid)* che prende in ingresso l'identificativo del gruppo.

```
static void printView(String vmsString)
{
    int gid = Integer.parseInt(vmsString);
    VMService.printView(gid);
}
```

Codice d'esempio 4.8 – Stampare una vista

4.2.8 **Ricerca su una vista**

Un peer dell'applicazione può selezionare dalla vista di un suo gruppo solo gli elementi che matchano con determinati vincoli. Per fare ciò invoca il metodo *printMatchingMembers()* che dopo aver creato un vettore che rappresenta i vicoli di ricerca ottiene gli elementi attraverso il metodo *VMService.getMatchingMembers(Vector c, int gid)* del VMS. Tale metodo restituisce in un vettore gli elementi che matchano con i vincoli specificati.

```
static void printMatchingMembers(String vmsString)
{
    int gid=Integer.parseInt(vmsString);
    Vector c=ConstraintFactory.createConstraint();
    Vector results=VMService.getMatchingMembers(c,gid);
    for(int i=0; i<results.size();i++)
    {
        VMSMember m=(VMSMember)results.get(i);
        System.out.println(i+"."+m.getPid());
    }
}
```

Codice d'esempio 4.9 – Ricerca sugli elementi di una vista

4.2.9 **Trasmissione e ricezione dei messaggi**

L'applicazione fornisce tre comandi principali per inviare i messaggi. Il primo permette di inviare un messaggio a tutti gli elementi di gruppo presenti nella propria località. Come possiamo vedere dal frammento di codice proposto è sufficiente recuperare la vista del gruppo con il metodo statico *VMService.getView(int gid)*, recuperare tutti gli elementi della vista con il metodo *getMembers()* e usare le primitive offerte dall'NMS per la trasmissione unicast (racchiuse nel metodo statico *NetUtil.send()*).

```
static void sendAllViewMembers(int gid, Message m)
{
     VMSView actualView = VMService.getView(gid);
     if(actualView != null)
     {
          Vector members = actualView.getMembers();
          for(int i=0; i<members.size(); i++)
          {
               VMSMember member = ((VMSMember)members.get(i));
               InetAddress addr = member.getAddress();
               NetUtil.send(m, addr, ReceiveMessage.REMOTE_PORT);
               System.out.println("Messaggio inviato!");
          }
     }
     else System.out.println("Gruppo inesistente");
}
```

Codice d'esempio 4.10 – Trasmissione agli elementi di una vista

Per trasmettere un messaggio agli elementi di una vista che soddisfano determinati vincoli, un peer dell'applicazione invoca l'operazione *sendMatchingMembers()* che recupera la lista di elementi interessati attraverso il metodo statico *VMService.getMatchingMembers(c, gid)* del VMS. Quest'ultimo metodo matcha tutti i profili degli elementi della vista del gruppo gid con i vincoli specificati e restituisce solo gli elementi che rispettano i vincoli.

```
static void sendMatchingMembers(int gid, Message msg, Vector c)
{
     Vector members = VMService.getMatchingMembers(c, gid);
     for(int i=0;i<members.size();i++)
     {
          VMSMember member = ((VMSMember)members.get(i));
          InetAddress address = member.getAddress();
          NetUtil.send(msg, address, ReceiveMessage.REMOTE_PORT);
          System.out.println("Messaggio inviato!");
```

```
      }
      else System.out.println("Gruppo inesistente");
}
```

Codice d'esempio 4.11 – Trasmissione agli elementi selezionati da una vista

Se un peer vuole inviare un messaggio ad un singolo elemento invoca il metodo *sendMember()* che recupera l'elemento dalla vista attraverso il metodo statico *VMService.getMember(gid,pid)* del VMS.

```
static void sendMember(int gid, int pid, Message message)
{
      VMSMember m = VMService.getMember(gid,pid);
      if(m!=null)
      {
            InetAddress address = m.getAddress();
            NetUtil.send(message,address,ReceiveMessage.REMOTE_PORT);
            System.out.println("Messaggio inviato!");
      }
      else System.out.println("Elemento inesistente");
}
```

Codice d'esempio 4.12 – Trasmissione ad un singolo elemento

La ricezione dei messaggi è effettuata dal ReceiveMessage. Quando è attivo, questo componente legge continuamente sulla porta di ingresso unicast sfruttando le primitive offerte dall'NMS. Quando riceve un messaggio effettua una semplice stampa.

4.3 **Testing**

Riportiamo questo paragrafo una serie di test effettuati sull'applicazione presentata al fine di misurarne l'affidabilità, l'occupazione di risorse e i tempi di reazione. In particolare mettiamo in evidenza l'utilizzo della memoria dato che tale applicazione è stata realizzata per essere utilizzata anche da dispositivi con limitate capacità.

4.3.1 **Occupazione memoria**

Abbiamo calcolato la dimensione degli oggetti usati dall'applicazione, quali il profilo di un elemento, un elemento memorizzato in una vista e una vista.

Un profilo creato con gli stessi attributi settati nel Codice d'esempio 4.1 (nome="Mario", cognome="Rossi",age=65, professione="pensionato") ha una dimensione di 1667 Bytes.
Tale profilo quando viene inviato in unicast in un oggetto MyProfileImpl che comprende anche gli identificativi GID/PID e l'indirizzo dell'elemento ha una dimensione di 1887 Bytes.
L'elemento contenuto nella vista, che comprende l'identificatore PID, l'indirizzo e il profilo, ha una dimensione di 1874 Bytes.
La dimensione di una vista dipende ovviamente dal numero di elementi in essa contenuti. Nella tabella 5.1 vediamo come esempi le dimensioni di viste che contengono da uno a dieci elementi.

Vista con 1 elemento	2161 Bytes
Vista con 2 elementi	2763 Bytes
Vista con 3 elementi	3365 Bytes
Vista con 4 elementi	3967 Bytes
Vista con 5 elementi	4569 Bytes
Vista con 6 elementi	5171 Bytes
Vista con 7 elementi	5773 Bytes
Vista con 8 elementi	6375 Bytes
Vista con 9 elementi	6977 Bytes
Vista con 10 elementi	7579 Bytes

Tabella 4.1 Dimensione vista

La dimensione di un messaggio che un peer può inviare agli elementi del gruppo, composto da gli identificativi GID/PID, da indirizzo del mittente e da una stringa di testo varia al variare della lunghezza dell'ultimo campo.
In conclusione riportiamo il footprint dell'applicazione, ovvero l'area di memoria e di disco richiesta per l'esecuzione.
La memoria fisica occupata dall'applicazione è di circa 350KB; questo valore rappresenta la dimensione di tutti i file class senza alcuna compressione. In particolare, possiamo notare che il valore della memoria fisica occupata dal VMS è di circa 26KB. Comprimendo tutti i file in una jar la memoria fisica occupata dall'intera applicazione è di circa 200KB.

Abbiamo inoltre notato che l'intera applicazione richiede run time un utilizzo di memoria circa 8MB; questo valore è stato calcolato misurando il valore della jvm per uno scenario applicativo in cui l'utente era associato a tre gruppi. Togliendo la memoria richiesta dalla jvm, l'applicazione utilizza circa 3,5MB e ovviamente questo valore varia a seconda del numero di gruppi a cui un elemento è associato principalmente per il fatto che varia il numero di viste gestite. In particolare un elemento che ha il ruolo di ME richiede un valore minore dal momento che non supporta tutte le operazioni di gestione dei gruppi.

CONCLUSIONI

Lo sviluppo di applicazioni collaborative avanzate in scenari MANET, caratterizzati da elevata mobilità dei terminali e da limitate risorse dei dispositivi degli utenti, solleva problemi complessi che richiedono l'adozione di nuove linee guida per il design e l'implementazione dei servizi.
La presente tesi introduce AGAPE, un sistema di gestione dei gruppi adatto allo sviluppo di servizi collaborativi avanzati in scenari MANET. In particolare ci siamo focalizzati sull'analisi, sullo sviluppo e sulla implementazione del View Manager Service (VMS) di AGAPE che consente la gestione delle viste dei gruppi. Il VMS fornisce ad ogni entità di AGAPE la completa visibilità dei membri associati al proprio gruppo e presenti nella stessa località di rete, insieme alle loro caratteristiche ed attributi. Infine, è stato mostrato il prototipo di una applicazione collaborativa cha consente lo scambio di messaggi tra utenti che appartengono ad uno stesso gruppo. Il prototipo implementato ha confermato che il supporto alla gestione dei gruppi fornito da AGAPE è adatto al supporto di applicazioni collaborative in scenari mobile ad-hoc network. In particolare, la completa visibilità dei membri colocati insieme ai loro attributi e caratteristiche facilita la collaborazione con utenti nuovi e precedentemente sconosciuti.
Questi primi risultati possano stimolare ulteriori attività di ricerca al fine di migliorare l'attuale prototipo e di verificare l'applicabilità sistema nello sviluppo di servizi collaborativi in diversi scenari applicativi.

Riferimenti:

[1] T.G. Zimmerman, Wireless networked devices: a new paradigm for computing and communication, IBM Systems Journal, 38(4), 1999.

[2] Sito web della Ericsson che contiene informazioni sul WCDMA: http://www.ericsson.com/wcdma

[3] R. Prakash and R. Baldoni, "Architecture for Group Communication in Mobile Systems", Symposium on Reliable Distributed Systems, West-Lafayette (IN), USA, 1998.

[4] M.S. Corson, J.P. Maker, and J.H. Cornicione Internet-based Mobile Ad Hoc Networking, *IEEE Internet Computing*, July-August 1999, pp.63-70;

[5] M. Conti and S. Giordano, Special issue on "Mobile Ad Hoc Networking", *Cluster Computing Journal*, 5 (2), April 2002;

[6] Sito web del Bluetooth Special Interest Group: http://www.bluetooth.com/

[7] C. Law, A.K. Mehta, and K.Y. Siu, A New Bluetooth Scatternet Formation Protocol, ACM/Kluver Mobile Networks and Applications Journal, Special Issue on Ad Hoc Networks, A.T. Campbell, M. Conti, and S. Giordano, Eds., 8(5), Oct. 2003.

[8] G. Zussman and A. Segall, Capacity Assignment in Bluetoot h Scatternets — Analysis and Algorithms, Proc. Networking 2002, LNCS 2345

[9] C.-C. Chiang, G. Pei, M. Gerla, and T.-W. Chen, Scalable routing strategies for ad hoc wireless networks, *IEEE Journal on Selected Areas in Communications*, 17, 1369–1379, 1999.

[10] M.S. Corson and A. Ephremides, A distributed routing algorithm for mobile wireless networks, *ACM/Baltzer Wireless Networks*, 1, 61–81, 1995

[11] C. E. Perkins and P. Bhagwat, “Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers,” *Comp. Commun. Rev.*, Oct. 1994, pp. 234–244.

[12] J. Broch, D. B. Johnson, and D. A. Maltz, “The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks,” IETF Internet draft, draft-ietfmanet-dsr-01.txt, Dec. 1998 (work in progress).

[13] Z.J. Haas and M.R. Pearlman, The Zone Routing Protocol (ZRP) for Ad Hoc Networks, Internet Draft, draft-ietf-manet-zone-zrp-02.txt, June 1999.

[14] S.H. Bae, S.J. Lee, W. Su, and M. Gerla, The Design, Implementation, and Performance Evaluation of the On-demand Multicast Routing Protocol in Multihop Wireless Networks, IEEE Network, Jan./Feb. 2000, pp.70–77.

[15] D. Bottazzi, A. Corradi, R. Montanari, "Context-Awareness for Impromptu Collaboration in MANETs".

[16] D. Bottazzi, A. Corradi, R. Montanari, "A Location-Aware Group Membership Middleware for Pervasive Computing Environments", *Proc. 8th Int. Symp. Computer and Communications (ISCC’03)*, Antalya, Turkey, June 2003.