Alma Mater Studiorum
Università degli Studi di Bologna

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea in Reti di Calcolatori

Comunicazione Context-Aware per Gruppi in Scenari Wireless Ad-Hoc

Candidato:
Gianpiero Napoli

Relatore:
Chiar.mo Prof. Antonio Corradi

Correlatore:
Dott. Ing. Dario Bottazzi

Anno Accademico 2004/2005 - Sessione I

Ai miei genitori, per avermi sostenuto
per tutti questi anni e ad Elisabetta:
senza di Lei non sarei mai arrivato
dove sono.

Indice

Introduzione

La crescente disponibilità di connettività wireless negli ambienti in cui gli utenti vivono e lavorano, la proliferazione dei dispositivi portatili, ed i recenti sviluppi delle tecnologie Mobile Ad-hoc NETwork (MANET) aprono un nuovo scenario. Nel nuovo scenario al fianco dei tradizionali servizi Internet quali il web e la mail, gli utenti richiedono la possibilità di fruire di servizi collaborativi avanzati. I nuovi servizi devono consentire la collaborazione impromptu fra utenti nuovi e precedentemente sconosciuti ovunque essi si trovino ed in qualunque momento. Esempi di servizi abilitati dalle tecnologie MANET sono costituiti da file sharing, il coordinamento di operatori in scenari di protezione civile o la condivisione di informazioni sul traffico fra diversi veicoli.

Le caratteristiche uniche delle Mobile Ad-hoc NETwork sollevano diversi problemi per lo sviluppo di servizi collaborativi. La topologia della rete non è determinabile a priori rendendo impossibili assunzioni sulle identità e caratteristiche 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 dalle richieste degli utenti di potere collaborare tramite dispositivi eterogenei sia per natura che per prestazioni quali lap-top, PDA o telefono cellulari.

La dinamicità delle MANET rende perciò inadatto l'utilizzo dei nomi per la scelta dei membri per la collaborazione e richiede l'adozione di nuove soluzioni che seguano strategie differenti. Basare la scelta dei partner per la collaborazione sulla base di informazioni di contesto quali la loro località, caratteristiche e attributi sembra un approccio adatto allo sviluppo di applicazioni in scenari Mobile Ad-hoc Network.

La tesi è organizzata come segue: il Capitolo 1 descrive il funzionamento delle reti wireless e fornisce dettagli sulle tecnologie su cui si basano. Il Capitolo 2 introduce le MANET e illustra i protocolli di routing che esse utilizzano. Nel terzo capitolo viene introdotto il framework AGAPE e i servizi che lo compongono. Nel quarto capitolo viene proposto un modello per il supporto alle comunicazioni di gruppo e vengono sviluppati due nuovi servizi per AGAPE: il Communication Service e il Binding Service. Nel quinto è ultimo capitolo viene presentata un applicazione che sfrutta i nuovi servizi realizzati. Infine è presente una piccola appendice che illustra l'installazione e la configurazione di AGAPE e dell'applicazione su PDA.

Capitolo 1

Tecnologie Wireless

1.1 Introduzione

La recente diffusione di dispositivi portabili che possono beneficiare di connessioni di tipo wireless e la presenza di connettività diffusa aprono un nuovo scenario in cui gli utenti non solo richiedono la possibilità di utilizzare servizi Internet, quali il WEB o la e-mail, ma richiedono di beneficiare ovunque si trovino, in ogni momento e anche in movimento di servizi collaborativi avanzati.
Nel nuovo scenario gli utenti richiedono di accedere alle funzionalità di rete attraverso dispositivi mobili ed eterogenei sia per natura che per capacità computazionali quali laptop, PDA e telefoni cellulari.
Un immediato vantaggio che deriva dall'utilizzo della tecnologia wireless riguarda la facilità d'installazione. Non essendo necessario installare cavi sono ideali anche per installazioni temporanee come mostre, fiere, congressi, situazioni di emergenza, gruppi di lavoro. In questo modo i dispositivi, e quindi gli utenti, possono spostarsi all'interno della rete rimanendo comunque connessi.
Esistono sul mercato molte proposte e soluzione per permettere a questi dispositivi di offrire agli utenti connettività wireless: Lo standard IEEE 802.11 e le sue varianti a,b,...,g, la tecnologia Bluetooth, le reti ad accesso radio: come il GSM, l'EDGE e il nuovo UMTS e la connessione attraverso raggi infrarossi IrDA.
La scelta della soluzione da utilizzare dipende da molti fattori. Ad esempio dal tipo di dispositivo utilizzato, dalle sue dimensioni, dalle sue capacità

computazionali e dal consumo delle batterie.
Per stabilire quale tecnologia è meglio utilizzare è importante classificare le reti e il loro utilizzo. Un' usuale classificazione, diffusa e accettata da tempo, usa come parametro la copertura della rete:

- **Body Area Network** o BAN;
- **Personal Area Network** o PAN;
- **Wireless Local Area Network** o WLAN;
- **Wireless Wide Area Network** o WWAN;

La figura 1.1 rappresenta la classificazione appena proposta:

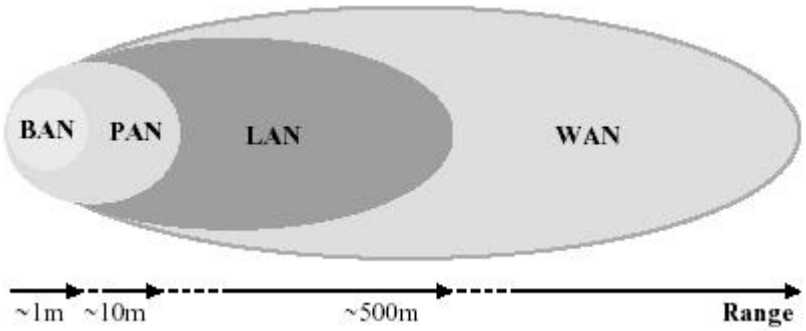

Figura 1.1: Classificazione delle MANET

1.1.1 Body Area Network

Le Body Area Network (figura 1.2) sono strettamente correlate con il concetto di computer distribuito sul corpo umano. Una rete BAN ha il compito di interconnettere i vari dispositivi eterogenei indossati dall'utente come ad esempio palmare,lettore MP3, auricolare, microfono e telefono cellulare.
La loro copertura è limitata alle dimensione del corpo umano quindi dell'ordine di 1-2 metri. La larghezza dell'ordine di qualche centinaio di Kbps. Il primo esempio che si ha delle reti BAN è un prototipo sviluppato da T.G. Zimmermann: questo prototipo permette lo scambio di dati tra i dispositivi indossati dall'utente utilizzando una piccola corrente che scorre attraverso la pelle umana.

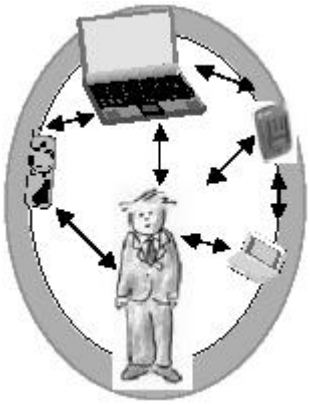

Figura 1.2: Rete BAN tra dispositivi indossati da un utente

1.1.2 Personal Area Network

Le Personal Area Network (figura 1.3) hanno il compito di interconnettere dispositivi vari nell'ambiente che circonda l'utente tipicamente in una stanza o in un ufficio. La copertura di questa tipologia di rete è quindi limitata alla decina di metri cioè le dimensioni della stanza o dell'ufficio in cui si trovano i dispositivi da connettere. La larghezza di banda delle Personal Area Network è dell'ordine di 1 Mbps. Un ottimo esempio per illustrare l'utilizzo di queste reti potrebbe essere il collegamento di dispositivi elettronici all'interno di una stanza: il pc portatile con il desktop o la macchina digitale che invia le fotografie alla stampante e il lettore MP3 che carica i brani preferiti direttamente dal computer.
La tecnologia che viene più frequentemente utilizzata per questo tipo di connessioni è il Bluetooth che opera nella frequenza libera di 2.4 GHz.

1.1.3 Wireless Local Area Network

Le Wireless Local Area Network hanno le stesse funzioni delle tradizionali reti LAN wired.
Come le tradizionali reti LAN, le WAN devono offrire alta capacità, piena connettività tra le stazioni che ne fanno parte e capacità di broadcast. Ma, a differenza delle reti cablate, devono tenere presente il problema del consumo energetico, il problema della mobilità dei dispositivi e la limitata larghezza di banda.

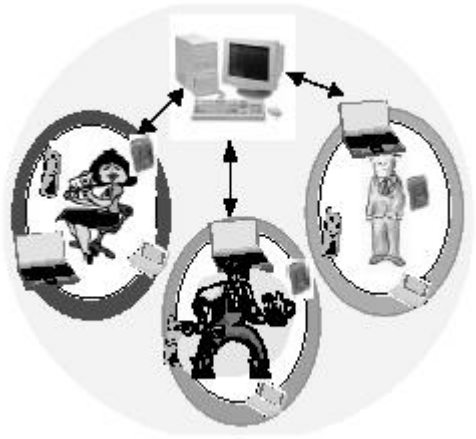

Figura 1.3: Rete PAN tra dispositivi che circondano l'utente

La copertura di una WAN è compresa tra i 100 e i 500 metri con una larghezza di banda che raggiunge la decina di Mega bit al secondo.
Attualmente la tecnologia maggiormente utilizzata per questo tipo di reti è lo standard IEEE 802.11b o 802.11g che permette di raggiungere rispettivamente una velocità di 11 e 54 Mega bit al secondo operando nella banda di 2.4 GHz.

1.1.4 Wide Wireless Area Network

Le Wide Wireless Area Network sono le reti wireless con copertura maggiore. Tali reti sfruttano la copertura della rete GSM usata per i telefoni cellulari. Per questo motivo gli utenti possono usufruire dei servizi offerti dalla rete internet ovunque ci sia copertura.
I servizi offerti da tali reti sono tipicamente l'uso della posta elettronica e la navigazione in internet.Gli standard più diffusi sono il GPRS e l'UMTS. Per quanto riguarda il GPRS, esso può usare alcune risorse del GSM con un bit rate massimo 171,2 Kbps. Il GPRS consentendo alle informazioni di essere trasmesse più velocemente tramite le reti mobili risulta essere un efficiente ed economico mezzo di trasmissioni dati mobile, riuscendo a sfruttare i protocolli TCP/IP e X25.

1.2 Modelli per reti wireless

Convenzionalmente si è soliti introdurre tre principali modelli di reti wireless:

- **Il modello a Cella**;
- **Il modello a Cella Virtuale**;
- **Il modello Ad-Hoc**.

1.2.1 Modello a Cella

In questo modello l'intera area in cui si vuole avere copertura di rete viene suddivisa in celle che possono anche sovrapporsi come illustrato in figura 1.4.

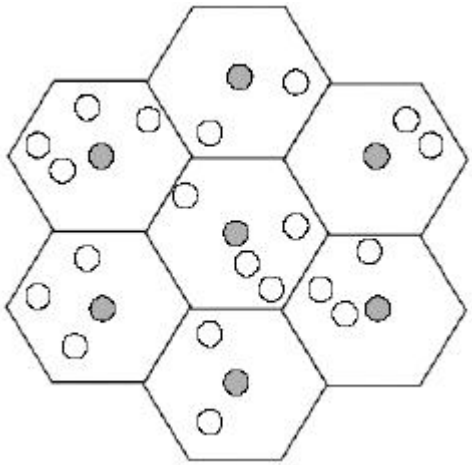

Figura 1.4: Divisione di una area in celle

Ogni cella possiede una base fissa detta *Base Station* (BS). Le Base Station delle diverse celle sono connesse tra loro tramite una rete cablata che si assume affidabile e dotata di una larga banda. I nodi mobili presenti nella rete sono detti *Mobile Host* (MH) e possono passare da una cella all'altra liberamente. Ogni Mobile Host per comunicare con un altro nodo del sistema utilizza sempre una Base Station.
Se un Mobile Host vuole comunicare con un altro Mobile Host della rete stabilisce un collegamento wireless con la Base Station della sua cella. Quest'ultima verifica se il destinatario è presente nella stessa cella e stabilisce un collegamento wireless con esso. Se invece il destinatario della comunicazione non si trova nella stessa cella la Base Station inoltra il messaggio alla Base

Station della cella in cui si trova il destinatario tramite un collegamento wired. Sarà quindi la nuova Base Station a recapitare il messaggio al Mobile Host destinatario.
I due casi sono illustrati nella figura 1.5

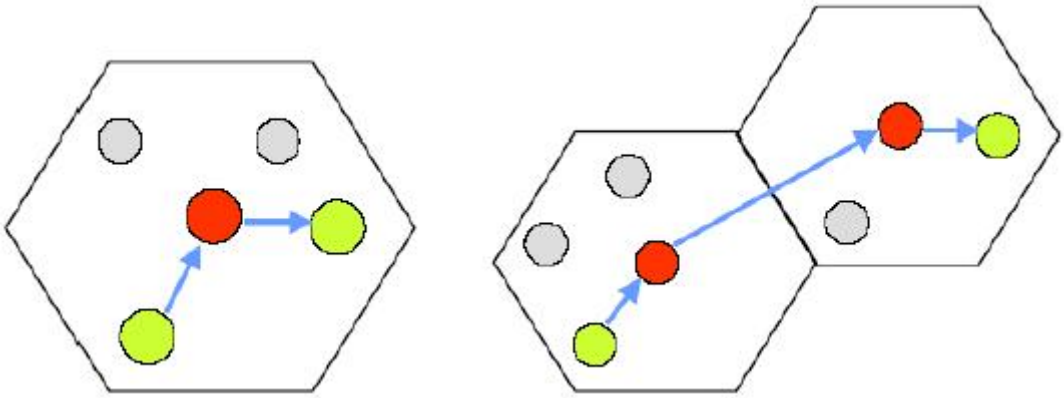

Figura 1.5: Comunicazione tra due Mobile Host nella stessa cella (a sinistra) e in celle diverse (a destra)

1.2.2 Modello a Cella Virtuale

Anche in questo modello lo spazio della rete viene partizionato in celle ma in questo caso anche le Base Station sono mobili e sono connesse tra loro tramite collegamenti wireless.
I collegamenti tra i Mobile Host sono coordinati dalle Base Station, ma non essendo più fissi la loro posizione e il loro grafo variano nel tempo (figura 1.6).
Se un Mobile Host deve spedire un messaggio ad un altro MH comunica il messaggio alla Base Station della cella in cui si trova. Quest'ultima invierà il messaggio al Mobile Host destinatario se esso si trova nella stessa cella. In caso contrario provvederà a trasferire il messaggio a un altra Base Station attraverso un collegamento wireless fino ad arrivare nella cella in cui è situato il destinatario.

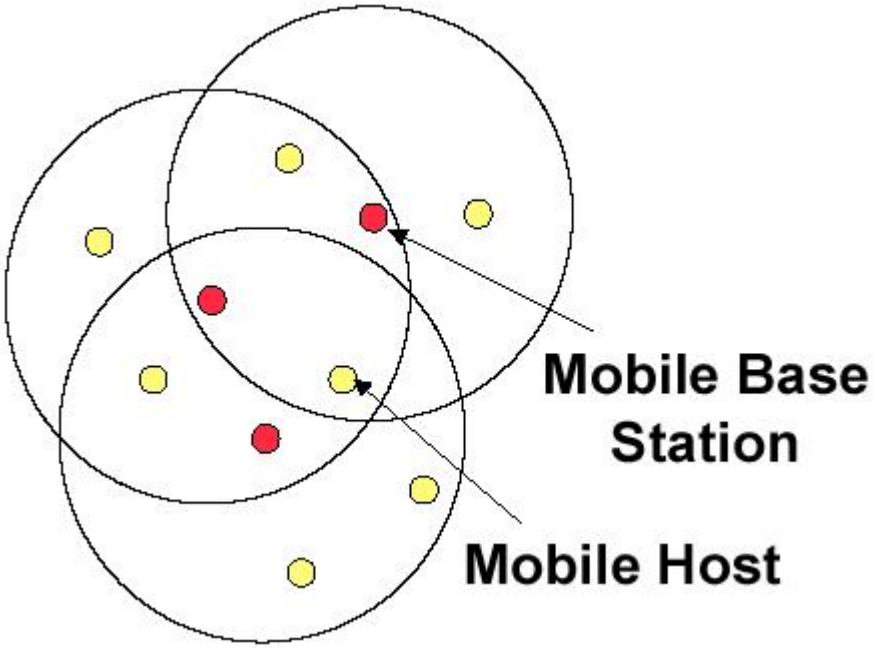

Figura 1.6: Grafo del modello a celle Virtuali

1.2.3 Modello Ad-Hoc

Rilassando ulteriormente il modello a cella virtuale si giunge al modello di rete Ad-Hoc. In questo modello viene a mancare la distinzione tra *Base Station* e*Mobile Host*.
Non essendo presenti nodi fissi o nodi mobili di riferimento, nella rete ad-hoc tutti i nodi devono prendere decisioni in maniera collettiva creando tutta una serie di problematiche nuove che non si trovavano nei precedenti modelli descritti.
A causa della frequente mobilità l'insieme di nodi vicini ad un host varia velocemente nel tempo. Gli spostamenti degli utenti non sono prevedibili e quindi la topologia della rete varia in modo arbitrario nel tempo portando a continue partizioni della rete e a successivi merge. Tutte le comunicazioni sono esclusivamente di tipo wireless.
Queste tipologie di rete sono adatte a scenari di utilizzo in cui non è presente alcuna infrastruttura di rete nè fissa nè mobile come scenari bellici e disastri improvvisi o semplicemente sale riunioni non attrezzate, areoporti o impianti sportivi. Si può immaginare una tassonomia delle reti Ad-Hoc in base alla

loro topologia che può essere *gerarchica (hierarchical) piatta (flat)*. In una

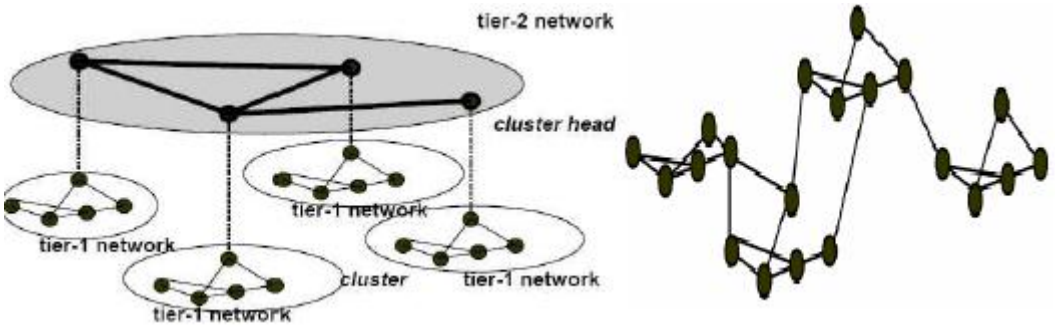

Figura 1.7: Topologia di una rete ad-hoc: gerarchica a sinistra e piatta a destra

rete gerarchica i nodi sono partizionati in gruppi detti cluster. Per ogni cluster è selezionato un cluster head attraverso i quali passa il traffico della rete. In una rete ad hoc piatta non è previsto nessun elemento di centralizzazione. Due nodi hanno la possibilità di entrare in comunicazione se la potenza del segnale è tale da permettere al nodo destinazione di sentire la trasmissione del vicino (ovvero se due nodi sono in copertura radio). Un vantaggio della rete piatta è la possibilità di stabilire più di un percorso tra nodo sorgente e destinazione; questo permette di valutare in modi diversi quale collegamento è da preferire, a seconda delle richieste e dell' utilizzo della rete. Il vantaggio della rete gerarchica è invece quello di minimizzare il numero delle informazioni di routing che vengono scambiate tra i nodi di uno stesso cluster e tra i cluster head.
In tutti i modelli descritti, c'è la possibilità che un nodo mobile possa muoversi fuori dal raggio di copertura degli altri nodi, diventando irraggiungibile e disconnettendosi dalla rete. Anche quando la rete è operativa e tutti i nodi sono raggiungibili, i sistemi mobili pongono comunque interessanti problematiche di gestione in quanto la topologia di rete può cambiare in qualsiasi momento favorendo fusioni, partizioni e merge di insiemi di nodi.

1.3 Tecnologie per la realizzazione delle MANET

Tra tutte le tecnologie disponibili per la realizzazione di collegamenti wireless ci concentriamo su due tecnologie particolarmente rilevanti:

- **Bluetooth**, per quanto riguarda le reti con raggio inferiore ai 10 metri;
- **IEEE 802.11**, per l'implementazione di reti single-hop ad hoc.

1.3.1 Lo Standard Bluetooth

Bluetooth è uno standard de facto creato da un gruppo di aziende riunite nel SIG (Special Interest Group). Lo standard 802.15 dello IEEE riprende una parte dello standard Bluetooth. Grazie a questa tecnologia è possibile integrare funzioni di comunicazioni dati e voce in un unico dispositivo in modo da abbassare notevolmente i costi e permettendo l'integrazione di questa tecnologia in molti dispositivi elettronici commerciali.

La rete bluetooth

In una rete realizzata con tecnologia bluetooth, una sola stazione assume il ruolo di master mentre le altre sono slave. Tutte le unità condividono lo stesso canale, il cui accesso è regolato dalla stazione master, formando una piconet.
Una piconet, rappresentata in figura 1.8, ha una bit rate di 1 Mb/sec ed è costituita da un master e da sette unità slave attive contemporaneamente. La piconet è il blocco fondamentale di una rete bluetooth. Se una unità partecipa contemporaneamente a due piconet sia ha la sovrapposizione e si forma una scatternet, rappresentata in figura 1.9.

Formazione di una piconet

Prima di poter iniziare una trasmissione di dati, l'unità bluetooth deve controllare che nessuna altra stazione stia operando nel suo spazio. Per fare ci l'unità entra in inquiry state.
In questa fase l'unità invia in continuazione un inquiry message. Un unità

puo rispondere a tale messaggio solo se sta ascoltando sul canale per trovare tale messaggio e solo se è sintonizzata su la frequenza di trasmissione utilizzata dall'unità trasmittente.
Dopo l'invio di tale messaggio e la ricezione delle eventuali risposte l'unità trasmittente conosce i dispositivi che gli sono vicini e a che frequenza lavorano. Per iniziare una comunicazione deve comunicare il suo indirizzo e il suo tempo di clock. Viene quindi avviata la routine di paging.
L'unità che comincia il paging è eletta automaticamente master della comunicazione e l'unita slave con cui si vuole comunicare è detta unità paged. L'unità paging invia un messaggio all'unità paged; a questo punto anche l'unità paged conosce l'indirizzo del master e può avere inizio la comunicazione.

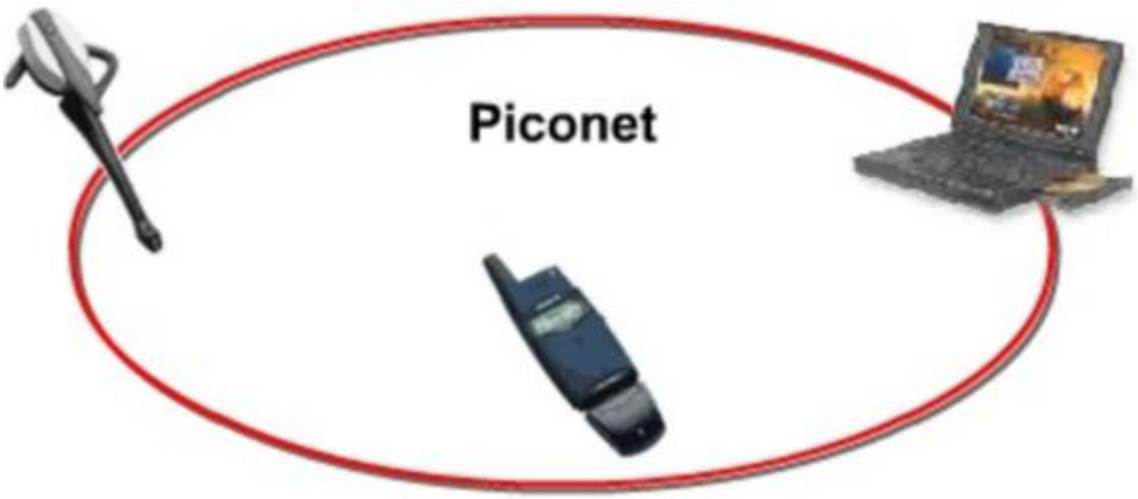

Figura 1.8: Formazione di una Piconet

Formazione di una scatternet

Una scatternet può essere costruita dinamicamente quando una o più unità partecipano a più piconet contemporaneamente. Il traffico da scambiare tra le due piconet passa sfruttando l'unità comune.
Attualmente lo standard Bluetooth contiene la nozione di scatternet ma non fornisce meccanismi per la sua realizzazione.

Figura 1.9: Formazione di una Scatternet

1.3.2 Lo standard IEEE 802.11

Il primo standard per le reti locali senza fili venne completato e reso noto dall'IEEE nel 1997. La prima versione dell'802.11 permetteva la creazioni di reti con data rate di 2Mb/sec utilizzando come strato fisico le onde radio a 2.4 GHz o gli infrarossi. Successivamente furono apportate delle modifiche al primo standard e vennero ottenute diverse estensioni tra cui:

- 802.11a, che opera nella banda 5 GHz e consente un data rate di 54 Mb/sec;
- 802.11b, che opera nella banda 2.4 GHz e consente un data rare di 11 Mb/sec.
- 802.11g, che opera nella stessa manda dell'802.11b, ed è quindi compatibile con esso, ma ha comunque un data rate di 54 Mb/sec

Altri gruppi lavorano su altre estensioni del protocollo cercando, ad esempio, soluzioni che permettano di avere elevata QoS per permettere la trasmissione di audio e video sulle reti 802.11 o di migliorare, rendendo più veloci, i protocolli già esistenti.

Architettura del protocollo 802.11

Lo standard 802.11 rappresentato in figura 1.10 specifica lo strato fisico e, all'interno dello strato data-link, il sottolivello di accesso al mezzo.

Per quanto riguarda lo stato fisico, 802.11 prevede la trasmissione o tramite infrarossi o tramite onde radio. Mentre per quanto riguarda il sottolivello di

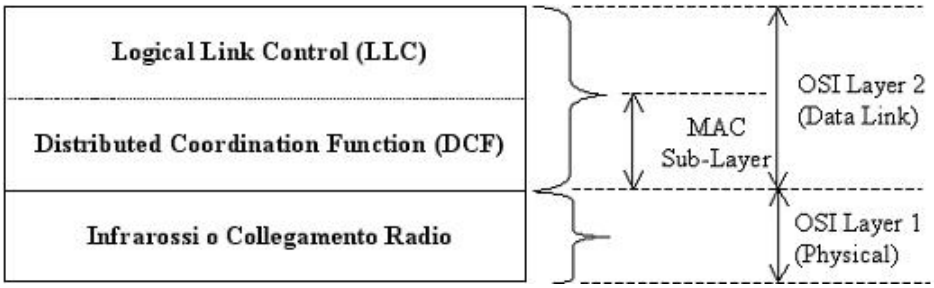

Figura 1.10: Architettura del protocollo 802.11

accesso al mezzo lo standard prevede e due protocolli, il *Distributed Coordination Function* (DCF) e il *Point Coordination Function* (PCF).
Il PCF funziona come un sistema a polling: un coordinatore permette l'accesso al mezzo fisico ad una sola stazione alla volta; chiaramente questo protocollo non può essere utilizzato in reti wireless che utilizzano il modello ad-hoc in quanto presuppone la presenza di un coordinatore centralizzato. La scelta ricade quindi sul primo protocollo citato.

Distributed Coordination Function

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 Distributed 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 rimarra 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 libero per un tempo superiore al DIFS. La stazione inizia a trasmettere quando il backoff timer viene azzerato.

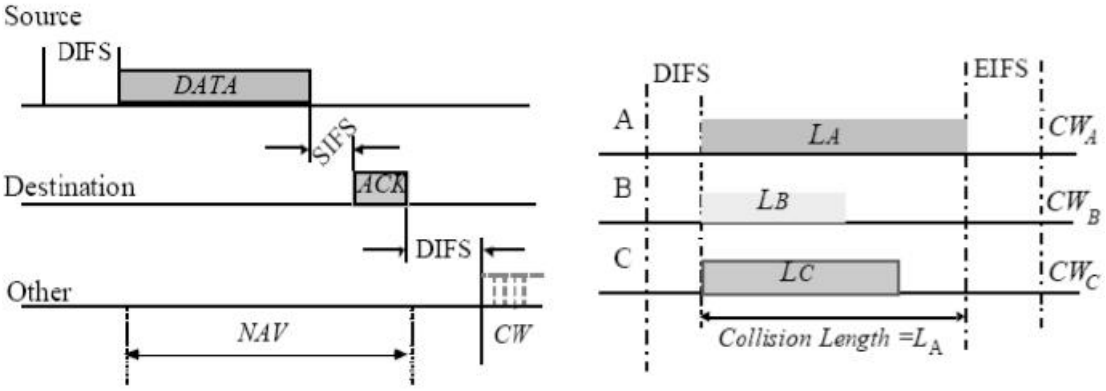

Figura 1.11: Trasmissione tramite DCF conclusa con successo (a sinistra) e con collisione (a destra)

Una rete basata sullo standard IEEE 802.11 può essere implementata utilizzano un'infrastruttura di supporto (mediante utilizzo di Access Point) o utilizzando il paradigma ad-hoc.
Grazie alla flessibilità del protocollo CSMA/CA è sufficiente che tutte le

stazioni siano sincronizzate per riuscire a lavorare in modalità ad-hoc. L'IEEE 802.11 usa due funzioni principali per sincronizzare le stazioni:

- **Acquisizione della sincronizzazione**: quando un terminale mobile vuole accedere ad una rete fa una scansione radio della banda alla ricerca di un particolare frame di controllo. Se tale frame viene trovato, allora la stazione si sincronizza altrimenti crea una nuova rete.
- **Mantenimento della sincronizzazione**: a causa della mancanza di una stazione centrale che fornisce un clock comune, viene trasmesso un segnale a intervalli regolari ad una frequenza nominale nota a tutti. L'intervallo di tempo è deciso dalla stazione che ha creato la rete.

Capitolo 2

Mobile Ad-hoc NETwork

2.1 Definizione di una rete MANET

Una Mobile Ad-Hoc NETwork (MANET) rappresenta un sistema di nodi mobili connessi attraverso tecnologia wireless. Tali nodi devono essere in grado di connettersi e comunicare tra loro senza alcuna infrastruttura di supporto preesistente.
Nel tempo sono stati sviluppate numerose soluzioni secondo il Modello Client/Server che hanno fornito meccanismi complessi che permettono l'invocazione di metodi su oggetti remoti in modo trasparente, come se fossero locali. Queste soluzioni no sono utilizzabili nello scenario delle reti Ad-Hoc in quanto le pritive che servono per chiamare procedure remote o per eseguire transazioni distribuite presuppongono la presenza di un canale di comunicazione affidabile, veloce e sempre disponibile. Lo stesso protocollo TCP risulta inadeguato all'interno di un contesto dinamico come le MANET. É inoltre sconsigliabile l'utilizzo di server centralizzati in quanto l'alta mobilità della rete e la sua inaffidabilita porterebbe alla disconnessione dei server e quindi verrebbe meno il servizio da loro messo a disposizione. I tradizionali metodi di interazione assumevano che tutte le entità interagenti conoscessero la localizzazione reciproca, fossero attive sempre tranne in condizioni di malfunzionamento e che potessero interagire con un protocollo di richiesta/risposta. Le MANET sono caratterizzate da topologie di rete dinamiche in cui non è possibile stabilire a priori i movimenti dei dispositivi che ne fanno parte; per questo è necessario un continuo interscambio dei nodi che fanno parte e costi-

tuiscono la rete in modo di mantenere le informazioni di rete continuamente aggiornate. Nasce quindi la necessità di trovare nuove soluzioni totalmente decentralizzate.
In questa nuovo paradigma di rete uno dei primi problemi che si deve affrontare è il problema del **routing**: Per permettere la comunicazione tra tutti i dispositivi appartenenti alla rete è necessario che ogni dispositivo sia in grado di effettuare il forwarding dei pacchetti. In questo modo anche chi non è in visibilità diretta può comunicare sfruttando adeguatamente il meccanismo di forwarding.
Il routing è il servizio fondamentale del livello rete il cui scopo è fare in modo che un insieme di stazioni sorgente possa comunicare al meglio con un insieme di stazioni destinazione non vicine, utilizzando al meglio le risorse presenti in rete.
Nel seguente paragrafo sarà trattato il routing nelle reti Ad-Hoc presentando alcuni protocolli attualmente utilizzati sia pre quanto riguarda la comunicazione punto-punto sia per la comunicazione di tipo punto-multipunto.

2.2 Routing nelle MANET

Una Mobile Ad-hoc NETwork può essere rappresentata attraverso un grafo in cui è indicato ogni nodo che ne fa parte e le sue connessioni con gli altri nodo.
Essendo la rete in questione una rete costituita da dispositivi che si muovono è naturale che tale grafo vari nel tempo, creando partizioni e merge di rete molto frequenti. E' inoltre importante sottolineare che, a differenza delle tradizionali reti cablate, i link tra due host spesso sono unidirezionali: infatti la rete è composta da dispositivi eterogenei la cui copertura è differente.
In questa tipologia di rete i protocolli di routing devono far fronte a numerose problematiche derivanti dalla natura stessa delle reti in oggetto:

- i dispositivi sono liberi di muoversi;
- hanno una copertura limitata e variabile da dispositivo a dispositivo;
- non tutti i dispositivi possono essere in diretta visibilità
- numerose partizioni e merge a causa di limitata autonomia dei dispositivi che partecipano alla rete e alla mobilità degli stessi;

- problemi del terminale nascosto e del terminale esposto

2.2.1 Classificazione dei protocolli di routing

I criteri con cui è possibile classificare i protocolli di routing per le reti Ad-Hoc sono molteplici.
Un primo criterio classifica i protocolli in base al momento in cui avviene l'elaborazione dei cammini.

- Nei protocolli *proattivi* vengono calcolati tutti i possibili percorsi senza sapere se poi verranno effettivamente utilizzati;
- Nei protocolli *reattivi* il calcolo del percorso di routing viene calcolato solo se è effetivamente richiesto: per questa ragione i protocolli reattivi sono anche detti *on-demand*;
- Nei protocolli *ibridi* il calcolo avviene in parte in maniera proattiva e in parte in modo reattivo: per i nodi vicini è possibile pianificare in anticipo i cammini, per i nodi lontani invece si utilizzano le funzionalità reattive; questa strategia è corretta se le applicazioni clienti del servizio di routing obbediscono al *principio di località*, cioè preferiscono per motivi di efficienza la comunicazione locale, con nodi vicini, a quella remota, con nodi distanti.

Nel resto di questo capitolo verranno esaminati i protocolli di routing Unicast e Multicast seguendo la prima classificazione proposta.

2.3 Ruting Unicast

Lo scopo delle trasmissioni unicast è l'invio di messaggi punto - punto da un nodo sorgente a un nodo destinatario.
Per ottenere questa trasmissioni ci sono due possibili scenari. Il primo scenario si ha quando sorgente e destinazione sono in diretta visibilità. Ci si trova nel secondo scenario quando due nodi che vogliono comunicare non si trovano nella condizione di comunicare direttamente: in questo caso è necessario passare attraverso altri nodi della rete affinchè un pacchetto giunga a destinazione. Lo scopo del routing è di calcolare il percorso che i pacchetti

devono seguire per arrivare a destinazione.
Ci sono due possibili approcci alla risoluzione del problema; il primo approccio prevede che ogni nodo abbia memorizzata una routing table locale in cui ha memorizzato per ogni nodo il next-hop attraverso il percorso; il secondo approccio prevede che in ogni pacchetto sia inserito anche il percorso completo.
Nel primo caso vengono calcolati tutti i possibili percorsi senza sapere se poi verranno effettivamente utilizzati. Nei protocolli reattivi il calcolo del percorso avviene solo se è effettivamente richiesto; per questo i protocolli appartenenti a questa categoria sono detti on-demand. I protocolli ibridi utilizzano le caratteristiche delle due famiglie precedenti.

2.3.1 Protocolli Proattivi

Questa tipologia di protocolli permette di avere a disposizione in ogni momento il percorso da una sorgente a una destinazione. Tutti i possibili percorsi di routing vengono infatti calcolati a priori senza sapere se verranno utilizzati.
In questo modo viene eliminato il ritardo quando un applicazione deve inviare un pacchetto.
Il vantaggio principale di questi protocolli è quello di garantire che un cammino verso una destinazione sia noto appena viene effettuata una richiesta al servizio di routing. Non sarà quindi presente il ritardo dovuto all'individuazione o al calcolo di un percorso per raggiungere la destinazione.
Questa caratteristica avvantaggia particolarmente la applicazioni che richiedono una bassa latenza come, ad esempio quelle interattive.

Destination-Sequenced Distance-Vector

Il protocollo proattivo DSDV adatta i protocolli di tipo Distance Vector alle reti MANET. I punti chiave introdotti da questo protocollo sono:

- Un meccanismo che permette di determinare l'età di un cammino;
- Aggiornamenti completi periodici e Aggiornamenti incrementali innescati dai cambiamenti topologici;

- Un ritardo introdotto sull'applicazione degli aggiornamenti ricevuti per i cammini instabili.

Il meccanismo che permette di determinare l'età di un cammino è basato su numeri di sequenza, assegnati e mantenuti da ogni nodo. Su ogni nodo il vettore delle distanze, oltre alle informazioni necessarie al routing, mantiene anche per ogni destinazione i relativi numeri di sequenza, inclusi negli aggiornamenti. Questo permette di distinguere se un cammino offerto sia più recente di uno già posseduto e, di conseguenza, se l'aggiornamento per una destinazione debba essere accettato o ignorato. Conoscere l'età di un cammino permette di evitare i loop e il fenomeno del counting-to-infinity presenti nei protocolli DV classici.
Per diminuire il traffico prodotto dalla trasmissione periodica degli interi vettori delle distanze viene introdotto il concetto di aggiornamento incrementale, trasmettendo solo le parti dei vettori che sono state interessate da variazioni della topologia.
Se una destinazione diventa irraggiungibile viene immediatamente diffusa un messaggio dal nodo che rileva il fallimento, mentre la presentazione di aggiornamenti riguardanti un nuovo cammino può essere ritardata per evitare di diffondere sulla rete delle notizie non affidabili. La stabilità di un cammino appena scoperto non è infatti certa e quindi il nodo ne ritarda la diffusione. Per implementare questo meccanismo di isteresi a ogni elemento della tabella di instradamento viene associato un setting time, cioè un tempo di assestamento trascorso il quale il cammino può essere considerato stabile e propagato tramite messaggi di aggiornamento, oppure scartato qualora il nodo dal quale la notizia è stata trasmessa non abbia nel frattempo ritrasmesso nulla che possa provare la sua prossimità.

Optimized Link State Routing

Il protocollo OLSR è di tipo LS ma è stato ottimizzato per lavorare in reti MANET.
Nei protocolli di tipo LS ogni nodo propaga lo stato dei propri link in *flodding*; in questo modo tutti i nodi che compongono la rete sono a conoscenza della nuova topologia di rete.

OLSR migliora le prestazioni del flooding attraverso l'uso di nodi intermedi nel processo di propagazione detti Multipoint Relay (MPR). Ogni nodo i seleziona indipendentemente dagli altri il proprio insieme di nodi MPR, denominato MPR(i). La generazione e la diffusione degli aggiornamenti della topologia riguarda i soli nodi MPR: risulta così diminuito il numero delle possibili implosioni, ossia diminuisce la probabilità che un nodo riceva due volte lo stesso messaggio.
Per mantenere aggiornata su ogni nodo la topologia della rete, ogni nodo eletto MPR trasmette in broadcast un messaggio Topology Control (TC). Ogni messaggio TC contiene l'insieme dei nodi raggiungibili dal nodo MPR che lo genera, insieme di nodi che include almeno l'insieme dei nodi che lo hanno scelto come MPR. I messaggi TC sono trattati in modo diverso, a seconda del tipo di nodo che li riceve: ogni nodo che riceve un messaggio TC aggiorna la topologia del proprio grafo di connessione, e i nodi MPR, in particolare ne proseguono la diffusione in modo che gli aggiornamenti siano propagati a tutta la rete. Dopo ogni aggiornamento topologico i nodi ricalcolano i cammini minimi e quindi aggiornano la propria tabella di instradamento.

2.3.2 Protocolli Reattivi

Come già accennato questa tipologia di protocolli per il routing, detta anche on-demand per ovvi motivi, calcolano in percorso verso un nodo destinazione solo nel momento in cui ci sono pacchetti da trasmettere.
Questo approccio riduce il traffico di rete risultando adatto in quei casi in cui, ad esempio, la banda è molto limitata; infatti non si avrà attività di routing se non ci sono dati da trasmettere.
In un protocollo reattivo possono essere evidenziate le seguenti fasi:

Route Discovery A seguito di una richiesta di trasmissione il servizio di routing inizia la ricerca della destinazione diffondendo un pacchetto di query. Quando tale pacchetto arriva alla destinazione o ad un nodo che sa come raggiungerla viene trasmessa alla sorgente la risposta.

Route Maintenance Costruito un cammino, a causa della mobilità e delle caratteristiche delle tecnologia, è necessario periodicamente controllarlo ed eventualmente ripararlo.

Route Deletion Se un cammino non viene utilizzato o se la fase di route maintenance ha avuto esito negativo le risorse precedentemente occupate sui nodi intermedi sono liberate.

Ad-hoc On Demand Distance Vector

Il protocollo AODV è il corrispettivo reattivo del protocollo DSDV. Come DSDV impedisce la formazione di cicli associando ad ogni cammino un numero di sequenza generato dal nodo destinazione; adotta per la manutenzione dei cammini un meccanismo di tipo event-driven basato su messaggi d'errore prodotti dai nodi che rilevano il fallimento.
In AODV tutti i link della MANET devono essere bidirezionali. Se il livello datalink consente link unidirezionali il corretto funzionamento di AODV non è garantito.
La tabella di instradamento è caratterizzata dai seguenti campi:

- Indirizzo IP della destinazione;
- Prefix Size della rete destinazione;
- Numero di sequenza della destinazione;
- Indirizzo IP del prossimo hop per raggiungere la destinazione;
- Tempo di vita della entry;
- Distanza in hop della destinazione;
- Interfaccia di rete da utilizzare per raggiungere la destinazione;
- Flag di stato attivo o inattivo.

Il *Prefix Size* permette di identificare una sottorete come un'unica entità AODV. Questo permette di rappresentare con unica entry le informazioni di instradamento comuni a tutti i nodi della sottorete e inoltre consente di eseguire una sola fase di discovery. Il tempo di vita è il tempo scaduto il quale la entry sarà rimossa se non viene usata per inoltrare pacchetti o non è aggiornata da AODV. Lo stato di attività è impiegato durante la fase di manutenzione.
AODV usa tre tipi di messaggi di controllo:

RREQ I messaggi di *Route REQuest* sono utilizzati nella fase iniziale di discovery.

RREP I messaggi di *Route REPlay* sono utilizzati nella fase finale di discovery.

RERR I messaggi di *Route ERRor* sono utilizzati nella fase di manutenzione.

Ogni indirizzo, in ogni messaggio di controllo, ha associato un numero di sequenza, il cui significato dipende dallo scopo dell'indirizzo e del messaggio. La fase di discovery ha inizio con la propagazione tramite flooding di una RREQ, illustrata in figura 2.1.

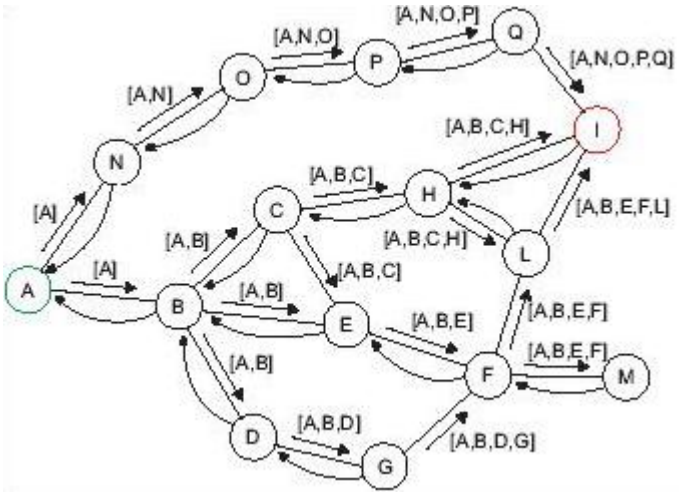

Figura 2.1: Propagazione di un messaggio RREQ nel protocollo AODV

Durante il flooding ogni nodo che ritrasmetta la RREQ aggiunge a questa il proprio indirizzo e numero di sequenza. Così facendo ogni nodo che riceva la RREQ può aggiornare la propria tabella con le *reverse route* o cammini inversi verso il mittente e verso i nodi che il messaggio RREQ ha attraversato. Utilizzando tali cammini inversi il nodo destinazione sarà in grado poi di eseguire, successivamente, la trasmissione della RREP al mittente. I cammini non utilizzati, creati durante il processo, saranno eliminati allo scadere del rispettivo tempo di vita.

Se il nodo che riceve la RREQ è la destinazione richiesta o un nodo che la possa raggiungere, allora esso genera una RREP, copiandovi dalla RREQ gli indirizzi dei nodi attraverso i quali la RREQ stessa è passata. La RREP è poi trasmessa in unicast verso l'origine, risalendo il percorso della precedente

RREQ: ogni nodo lungo la reverse route attraversata dalla RREP, aggiornerà la propria tabella con la nuova destinazione e inoltre con tutte le entry relative ai nodi situati tra il nodo stesso e la destinazione, costruendo la *forward route*, o cammino in avanti, verso la destinazione, come illustrato in figura 2.2.

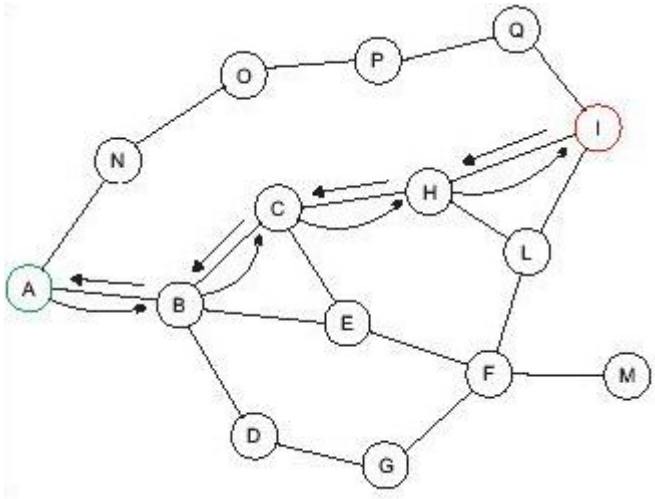

Figura 2.2: Propagazione di un messaggio RREP nel protocollo AODV

Inoltre se il nodo che genera la RREP non è la destinazione allora il nodo trasmette verso la destinazione una ulteriore RREP (detta *gratuitous RREP*) per consentirgli di raggiungere il nodo sorgente della RREQ, evitando così un ulteriore probabile fase di discovery iniziata dalla destinazione.
La tabella di un nodo può venire aggiornata sia in seguito alla ricezione di una RREQ, sia dopo la ricezione di una RREP; se una entry è già presente nella tabella, AODV l'aggiorna se si verifica uno dei seguenti fatti:

1. Il cammino scoperto è più nuovo di quello attualmente usato; ciò è determinabile tramite i numeri di sequenza associati ad ogni indirizzo.
2. Il numero di sequenza è lo stesso, ma l'entry non è attiva: in tal caso viene riportata in stato attivo.
3. A parità di numero di sequenza della destinazione il numero di hop del nuovo cammino è inferiore.

Si nota quindi che vengono privilegiati i cammini più nuovi, anche se più lunghi, a quelli di distanza minore, ma meno recenti.

Ogni nodo controlla il funzionamento dei propri link tramite la ricezione di beacon HELLO trasmessi dai vicini. Se un nodo rileva che un link non è più funzionante disattiva tutti i cammini che da questo dipendono e invia in brodcast locale, su tutte le interfacce AODV, un messaggio d'errore RERR per avvertire i vicini di tutte le destinazioni che non risultano più raggiungibili a causa del fallimento del link. La procedura di disattivazione è ricorsiva: ogni nodo che riceva una RERR pone in stato inattivo le entry della propria tabella la cui destinazione sia tra quelle dichiarate irraggiungibili e il prossimo hop sia l'indirizzo d'origine della RERR: se un nodo pone delle entry in stato inattivo ritrasmette a sua volta una RERR con le destinazioni disattivate. Tale processo di disattivazione prosegue fino ai nodi iniziali dei cammini.
Un cammino corrispondente ad un'entry posta in *stato inattivo* non può essere usato per inoltrare pacchetti e se non torna in stato attivo viene successivamente rimosso. Le entry sono poste in stato inattivo piuttosto che cancellate perchè altrimenti molte informazioni di stato, come i numeri di sequenza, andrebbero perdute e non sarebbe possibile valutare adeguatamente le RREQ e RREP che giungano al nodo.
AODV evita il problema del *broadcast storm* nella fase di propagazione dei messaggi RREQ identificando ogni RREQ e adottando una cache per essi su ogni nodo. Le RREQ sono identificate attraverso l'indirizzo dell'origine e il valore di un contatore sul nodo d'origine stesso. Questa soluzione non risolve però il problema dell'implosione visto in **??**, perchè non evita la trasmissione multipla di una RREQ verso un nodo.

Associativity Based Routing

ABR è un protocollo di routing reattivo sviluppato per reti mobili ad-hoc. L'idea chiave di ABR è utilizzare come metrìca di scelta dei cammini la stabilità nel tempo e nello spazio piuttosto che la lunghezza: un cammino usato da più tempo è preferito ad uno appena creato anche se questo fosse più corto. Questo criterio permette al protocollo di utilizzare i cammini più stabili, cioè quelli che richiedono minori interventi di manutenzione.
Per stimare il tempo di vita di un cammino ABR utilizza sia misure dirette come l'energia residua della stazione o l'intensità del segnale rivevuto dai

nodi adiacenti sia una propria metrica detta associatività, il cui scopo è catturare il grado di stabilità dei link verso i nodi vicini.
Per misurare l'*associatività* ogni nodo trasmette periodicamente un beacon e conta il numero di beacon ricevuti da ogni nodo vicino. Se il beacon di un nodo non viene ricevuto per un tempo prefissato il relativo contatore viene azzerato. Dati i nodi i, j e k se in i il valore del contatore corrispondente a j è maggiore del valore di quello di k allora il link (j, i) è più stabile del link (k, i).
Come altri protocolli reattivi ABR si compone di tre fasi: una fase di Route Discovery, una fase di Route Mainteneance e una di Route Deletion. L'origine inizia la fase di discovery diffondendo mediante flooding una *Broadcast Query* (BQ). Durante la fase di propagazione ogni nodo intermedio inserisce nella BQ ricevuta il proprio identificatore e un valore che stima la qualità del link verso il nodo dal quale è stata ricevuta e quindi continua la propagazione. I parametri di qualità inseriti comprendono anche il valore del contatore di associatività sull'ultimo hop attraversato. La destinazione, ricevuta la prima BQ, attende ulteriori BQ per un tempo prefissato: allo scadere del tempo sceglie tra le BQ ricevute quella che riporta i migliori parametri di qualità e trasmette quindi la risposta verso il nodo sorgente lungo il cammino seguito dalla BQ selezionata. I nodi lungo il percorso seguito dalla risposta attivano la *forward route* verso la destinazione.
Costruito un cammino bidirezionale da sorgente a destinazione inizia la fase di mantenimento. Durante questa fase i nodi lungo il cammino che rilevino il fallimento di un link verso la destinazione tentano prima autonomamente di ripararlo, innescando una nuova fase di discovery, schema noto anche come *salvaging*. Se il salvaging fallisce trasmettono verso il nodo sorgente un messaggio di notifica dell'errore. Quando questo arriva al nodo sorgente esso inizia una nuova fase di discovery. Se anche questa non ha successo il nodo sorgente notifica il fallimento alle entità del livello di trasporto.
Se la fase di mantenimento termina con insuccesso allora ha inizio la fase di cancellazione. ABR propone due alternative. Nella prima il nodo sorgente trasmette in flooding un messaggio di controllo che invita i nodi del cammino che lo ricevano a cancellare le entry dalla propria tabella di instradamento. Nella seconda, meno costosa, ogni entry ha un tempo di vita e se non vengono

utilizzate per instradare traffico i nodi le eliminano.

2.3.3 Protocolli Ibridi

I protocolli ibridi cercano di combinare i vantaggi di entrambe le famiglie citate precedentemente.
Il calcolo del cammino di routing avviene in parte in maniera proattiva e in parte in modo reattivo: per i nodi vicini è possibile pianificare in anticipo i cammini, per i nodi lontani invece si utilizzano le funzionalità reattive; questa strategia è corretta se le applicazioni clienti del servizio di routing obbediscono al principio di località, cioè preferiscono per motivi di efficienza la comunicazione locale, con nodi vicini, a quella remota, con nodi distanti.

Zone Routing Protocol

ZRP è un protocollo ibrido il cui obiettivo è unire i vantaggi del routing reattivo a quelli routing proattivo.
A tale scopo ZRP introduce il concetto di zona. Una *zona* $Z(k, n)$ di raggio k centrata sul nodo n è definita come l'insieme dei nodi distanti da n non più di k hop:

$$Z(k, n) = \{i : nodo \mid H(n, i) \leq k\}$$

dove $H(n, i)$ è la distanza del nodo i dal nodo n. Il nodo n è detto il *nodo centrale* della zona, mentre i nodi b tali che $H(n, b) = k$ sono detti i *nodi periferici* o *nodi di bordo* di n (vedi figura 2.3).

Il valore di k deve essere piccolo rispetto al diametro della rete. ZRP è organizzato in quattro componenti fondamentali:

IARP L'IntrAzone Routing Protocol è il protocollo proattivo che opera all'interno di una zona e il cui scopo è servire le richieste di routing da un nodo della zona ad un altro.

IERP L'IntErzone Routing Protocol è il protocollo reattivo utilizzato per servire le richieste di routing verso nodi esterni alla zona.

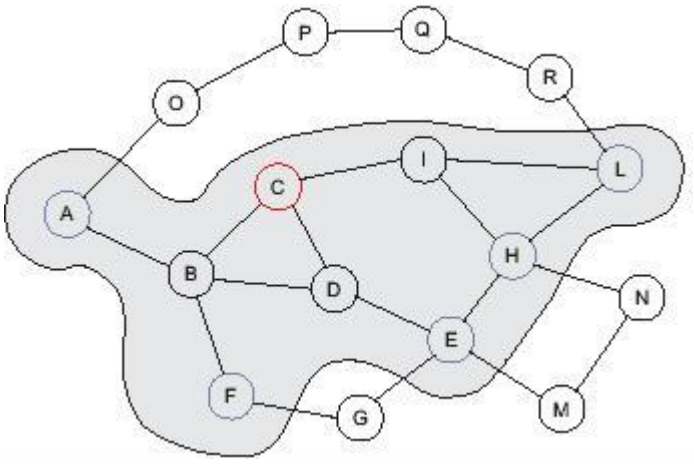

Figura 2.3: Esempio di una zona nel protocollo ZRP

BRP Il Bordercast Protocol è un protocollo di flooding selettivo che che si appoggia all'IARP e che permette ad un nodo di effettuare una trasmissione multicast verso i nodi periferici della zona.

NDP Il Neighbour Discovery/Maintanance Protocol si interfaccia alle funzioni di livello datalink per rilevare i vicini di un nodo.

Per l'implementazione di IARP è possibile utilizzare un qualunque protocollo di routing proattivo, purchè gli aggiornamenti topologici non superino la zona in cui il protocollo opera. IARP utilizza NDP per determinare i vicini di un nodo.

La fase di Route Discovery ha inizio se il nodo richiesto non è raggiungibile tramite IARP, in quanto non appartiene alla zona. Il nodo inizia allora una ricerca tramite IERP: i messaggi di query IERP sono trasmessi ai nodi periferici di una zona utilizzando BRP. Quando un pacchetto di query arriva su un nodo periferico questo controlla se il nodo richiesto appartenga ad una delle zone di cui è nodo di bordo. Se la destinazione richiesta è raggiungibile il nodo risponderà alla richiesta, altrimenti la ritrasmetterà, utilizzando BRP, verso i nodi di bordo delle zone a cui appartiene. Il cammino verso la destinazione può essere accumulato o nel pacchetto di query, come in DSR o, per diminuire il carico sulla rete, nella risposta.

I pacchetti ordinari sono poi inoltrati sul cammino verso una destinazione fuori zona tramite *loose source routing* sui nodi di bordo e tramite IARP tra i nodi di bordo di una zona.

La Route Mainteneance è eseguita sui cammini tra nodi di zone diverse nel momento in cui sia necessario ripararli. A tale fine viene usata una procedura di riparazione locale, simile al salvaging di ABR, che ripristina il cammino riconnettendo nodi periferici vicini in grado di raggiungere il nodo sorgente e quello destinazione.
In ZRP non esiste alcun coordinamento tra le zone, che possono quindi sovrapporsi. Può accadere che un nodo sia membro o nodo di bordo di numerose zone: questo può causare un peggioramento delle prestazioni di BRP, tanto da rendere preferibile l'uso del flooding anche all'interno delle zone.

Location Aided Routing

Quando sono disponibili informazioni di posizione e velocità è possibile ottenere un servizio di routing decisamente più efficiente dei protocolli precedentemente visti.
LAR è un protocollo reattivo che sfrutta informazioni di posizione e velocità di un nodo, precedentemente acquisite, per stimarne la posizione attuale e "orientare" la fase di Route Discovery. LAR ricorre al flooding solo se non vi sono dati precedenti di posizione: in questo modo diminuisce il traffico dovuto ai pacchetti di controllo del protocollo.
Ogni nodo rileva e memorizza continuamente la posizione e la velocità dei propri vicini oltre al tempo al quale l'acquisizione è avvenuta. L'inoltro dei pacchetti ordinari è table-driven. Quando un nodo destinazione D non è presente nella tabella del nodo sorgente S, LAR inizia la fase di discovery.
Se precedentemente il nodo D era stato vicino di S allora questo conoscerà i dati di posizione e velocità di D al tempo dell'acquisizione t_0. In base a questi parametri S calcola la *expected zone*, o zona attesa, entro la quale è probabile che il nodo D si trovi al tempo t_1. La *zona attesa* è una sfera centrata sulla posizione del nodo al tempo t_0 e il cui raggio dipende dalla velocità che il nodo aveva in t_0.
LAR prevede due diversi schemi di discovery. Nel primo la zona in cui la query viene propagata, detta *request zone*, è il più piccolo parallelepipedo che include sia S che la zona attesa di D: la query non può uscire dalla request zone durante il flooding. Le informazioni geometriche della request zone stessa sono incluse nella query in modo che i nodi che la ricevano pos-

sano determinare se propagarla o meno.
Nel secondo schema ogni query contiene la posizione stimata di D e la distanza stimata $Dist$ del nodo che ritrasmette la query dal nodo D stesso. Sia D_i la distanza di i dalla destinazione D. Quando il nodo sorgente S crea il pacchetto di query pone $Dist = D_S$. Se un nodo intermedio i ritrasmette la query pone $Dist = D_i$. Quando un nodo j riceve la query per D dal nodo i la ritrasmette solo se:

$$Dist + \delta > D_j$$

cioè solo se j è più vicino a i almeno della lunghezza δ. Tale valore è necessario per tenere conto degli errori di posizione, soprattutto se i e j sono vicini.

2.4 Routing Multicast

Il multicasting consiste nell'invio di messaggi punto - multipunto cioè da un nodo a un gruppo di nodi identificati da un unico indirizzo di destinazione. I membri di un gruppo sono dinamici e possono unirsi o lasciare il gruppo in qualsiasi momento. Inoltre un nodo pu far parte di più gruppi contemporaneamente e può mandare pacchetti anche a gruppi a cui non appartiene. I protocolli maggiormente utilizzati nelle reti cablate non sono utilizzabili nelle MANET a causa di diversi fattori: banda limitata, bassa potenza di calcolo, assenza di infrastrutture e autonomia energetica limitata. Questi fattori uniti alla dinamicità della topologia della rete obbligano le reti ad-hoc ad utilizzare dei protocolli che permettano di creare dinamicamente dei percorsi per collegare i nodi quando questi ne hanno bisogno.

Multicast Ad hoc On demand Distance Vector

MAODV è un protocollo di routing multicast, estensione di AODV visto precedentemente. La struttura di distribuzione che collega i router multicast di un gruppo forma un albero: quando un router riceve un pacchetto destinato al gruppo lo inoltra a tutti i router vicini di quel gruppo, ad eccezione del nodo origine del pacchetto.

Ogni nodo, oltre alla tabella di instradamento unicast di AODV, ha una tabella di instradamento multicast che comprende per ogni gruppo le seguenti informazioni:

- L'indirizzo IP del gruppo
- Il numero di sequenza del gruppo
- Il numero di hop per raggiungere il primo membro upstream del gruppo
- L'indirizzo IP del leader del gruppo
- Il numero di hop per raggiungere il leader
- Un elenco dei nodi vicini che permettono di giungere ai membri del gruppo; ognuno di tali vicini è caratterizzato dalle seguenti ulteriori informazioni:
  - L'indirizzo IP del nodo vicino
  - La direzione del link
  - Un flag di attivazione del link

Il *group leader* è in genere il primo nodo che è divenuto membro del gruppo. Ad esso spettano diversi compiti di manutenzione compresa l'assegnazione e la diffusione del numero di sequenza.
La direzione di un link può essere *upstream* o *downstream*, a seconda che rispettivamente il group leader sia o non sia raggiungibile attraverso di esso.

Route Discovery
Quando un nodo S vuole divenire membro di un gruppo G diffonde mediante flooding una Route REQuest o RREQ simile a quella di AODV .
La destinazione della RREQ è un qualunque nodo membro di G. I nodi non membri che inoltrano la RREQ costruiscono, come in AODV, le reverse route verso S e inoltre aggiornano la loro tabella multicast, aggiungendo S ai nodi membri di G in direzione downstream: il link verso S non viene però attivato.
Quando la RREQ di S arriva ad un nodo D membro di G, questo trasmette ad S una RREP, estesa con alcune informazioni di gruppo, tra le quali l'indirizzo del group leader e la distanza di D da questo. La RREP utilizza la reverse route precedentemente costruita dalla RREQ per raggiungere S e costruisce la forward route da S a D.
Ogni nodo intermedio attraversato da una RREP aggiorna la propria tabella multicast, aggiungendo un link upstream per il gruppo G verso D: se un link

upstream per G è già presente questo viene aggiornato solo se il numero di sequenza di G nella RREP è più recente di quello del nodo o se a parità di questo il numero di hop per raggiungere il nodo membro di G è inferiore.
L'origine S attende per un tempo predeterminato l'arrivo delle RREP. Quando S riceve una RREP aggiorna le proprie tabelle unicast e multicast verso i nodi membri come i precedenti nodi intermedi. Trascorso il periodo di attesa, il nodo S, ricevuta almeno una RREP, attiva il link upstream verso il gruppo G, trasmettendo verso il nodo membro prescelto un messaggio MACT.

Route Maintenance

MAODV prevede appositi meccanismi di manutenzione della struttura di distribuzione ad albero di un gruppo. Quando un router multicast R perde il link upstream verso un gruppo G, entra in fase di manutenzione e cerca di ripristinare il link upstream eseguendo una Route Discovery secondo le modalità precedentemente illustrate.
Se però R avesse dei link downstream verso altri nodi di G questi potrebbero rispondere alle RREQ generando dei cicli. Per evitare tale possibilità la RREQ diffusa da R contiene anche la distanza di questo dal group leader precedente al guasto. In tal modo i nodi downstream di R non rispondono ai messaggi RREQ di R. Ripristinato il link upstream verso G, i nodi del sottoalbero, precedentemente partizionato aggiornano le loro distanze dal group leader.
Se R fallisce nel riparare il link upstream per G allora questo e i nodi downstream risulteranno separati da G. In questo caso viene scelto un nuovo group leader tra i nodi membri appartenenti al sottoalbero che formerà così una nuova partizione di G. Se successivamente il group leader della nuova partizione rilevasse i beacon trasmessi da un altro group leader inizierebbero le operazioni necessarie al ricongiungimento delle due partizioni.

On Demand Multicast Routing Protocol

ODMRP è un protocollo di routing multicast per reti mobili ad-hoc basato su una struttura di distribuzione a *mesh*.
Per ogni gruppo ODMRP individua quattro tipologie di nodi: i sender, i receiver, i forwarder e il resto dei nodi. L'obiettivo di ODMRP è quello di creare

una rete di distribuzione aciclica, orientata dai *sender*, le radici della mesh, ai *receiver*, le foglie. I nodi intermedi che la mesh attraversa sono i *forwarder* e l'insieme di questi forma il *forwarding group* che connette i sender ai receiver. In ODMRP la fase di Route Discovery è detta pertanto *forwarding group setup*.
ODMRP non prevede messaggi di join e leave e neppure dei meccanismi per la manutenzione della struttura di distribuzione: questa viene infatti periodicamente tenuta aggiornata tramite messaggi di *advertisement* generati dai sender e dai receiver che vogliono essere membri del gruppo. Il periodo di tale advertisement non è costante e dipende dalla mobilità dei nodi della mesh. ODMRP diminuisce l'overhead causato da tale advertisement individuando dei cluster di nodi.

Forwarding Group Setup
Quando un nodo S ha dei dati destinati ad un gruppo G inizia a diffondere periodicamente tramite flooding una *Join Query* segnalando così ai nodi a lui vicini di voler raggiungere i receiver di G. I nodi che diffondono la Join Query aggiornano anche le proprie tabelle di instradamento, costruendo le Reverse Route verso il nodo S.
Un receiver R del gruppo G può ricevere Join Query da più nodi sender. R registra tutte le Join Query ricevute e eventualmente ne prosegue il flooding per permettere ad altri nodi receiver di riceverle. Periodicamente poi R trasmette in broadcast un messaggio *Join Reply* che contiene tutte le coppie selezionate (sender, previous-hop).
Quando un nodo riceve una Join Reply controlla se il proprio indirizzo compare o meno tra i previous-hop delle coppie presenti nella Join Reply. Se è presente allora il nodo diviene un forwarder e di conseguenza aggiorna la propria tabella, inserendo una nuova entry per il gruppo verso il nodo che ha trasmesso la Join Reply. Il nodo estrae poi dalla Join Replay tutti i sender del gruppo per i quali deve divenire un forwarder, cerca nella propria tabella i previous-hop per quei sender e infine trasmette in broadcast una nuova Join Reply. In questo modo la propagazione delle Join Reply prosegue verso i sender e non interessa nodi non scelti dai receiver.
Quando un sender riceve una Join Reply controlla se tra le coppie con lo

stesso (sender, previous-hop) compare il proprio indirizzo: se così è allora il nodo dal quale proviene la Join Reply porta a dei receiver del gruppo e di conseguenza aggiorna la propria tabella. A questo punto il sender può iniziare a trasmettere i dati destinati al gruppo: finchè ci sono dati da trasmettere provvederà periodicamente al refresh del forwarding group tramite le Join Query. Quando non avrà più dati la trasmissione periodica delle Join Query cesserà e le entry relative al sender nel forwarding group scadranno e saranno cancellate.

Predizione della mobilità

Il flooding periodico di Join Query è necessario a ODMRP per mantenere la struttura di distribuzione a mesh il più possibile aggiornata: se infatti la mobilità fosse eccessiva i sender a causa del fallimento di un link potrebbero non essere in grado di raggiungere i receiver. Per contro se la mobilità non è alta, ma la frequenza di aggiornamento è troppo elevata allora le Join Query diminuiscono inutilmente la banda disponibile per la comunicazione, causano collisioni e congestioni sulla rete e inoltre sprecano energia.
É quindi necessario adattare la frequenza del flooding alla mobilità dei nodi che formano la mesh. Per ottenere questo ODMRP stima il tempo per il quale due nodi vicini resteranno in visibilità. Una stima abbastanza attendibile di tale tempo è ottenibile ad esempio utilizzando i dati di posizione e velocità acquisiti tramite ricevitori GPS che equipaggino i nodi. Attraverso la durata stimata di un link è poi possibile stimare la durata di un cammino da sender a receiver e quindi determinare il tempo entro il quale sarà opportuno ritrasmettere una Join Query.

Broadcast gerarchico

Per ridurre l'overhead introdotto dal flooding gli autori di ODMRP propongono un protocollo di broadcast gerarchico. Ogni cluster è composto da tre tipi di nodi:

cluster head (CH) É il nodo centrale di un cluster: la sua portata massima in trasmissione definisce il confine del cluster.

gateway node (GN) É un elemento dell'insieme dei nodi che stanno sul

confine di un cluster: un nodo è un gateway se "vede" almeno due cluster head.

ordinary node É ogni nodo che non è un cluster head o un gateway.

Il protocollo ideato è passivo, in quanto non ha propri messaggi di controllo e le informazioni sono propagate in piggyback sui messaggi propri di ODMRP. Per tale ragione il protocollo è chiamato *Passive Clustering* (PC).
Inizialmente la rete non è partizionata: solo successivamente, man mano che i messaggi di controllo di ODMRP sono trasmessi si formano i cluster nei quali il solo CH e i GN possono propagare le Join Query. I nodi ordinari non saranno mai, almeno finchè restano tali, dei forwarder e quindi o ascolteranno le Join Query trasmesse dal rispettivo CH e GN, se sono dei receiver, o inizieranno la diffusione di una Join Query, non propagata dai nodi ordinari, nel momento in cui dovessero divenire dei sender.

Capitolo 3

AGAPE: Allocation and Group-Aware Pervasive Environment

Il nuovo scenario che si viene a creare grazie all'enorme disponibilità di dispositivi wireless a basso costo, permette agli utenti di stabilire collaborazioni in modo ovunque e in qualunque momento.
In questo nuovo scenario utenti con gli stessi interessi che vogliono perseguire un obiettivo comune hanno la necessita di creare dei gruppi di lavoro e comunicare.
La natura asincrona e inaffidabile degli ambienti MANET portano a riconsiderare i tradizionali metodi di comunicazione di gruppo correntemente utilizzati.
In una rete Ad-Hoc mobile gli utenti appaiono e scompaiono in continuazione in maniera imprevedibile; essi cambiano la loro posizione nella rete e quindi la topologia della rete stessa varia in continuazione. Inoltre l'eterogeneità dei dispositivi di cui gli utenti dispongono porta la necessità di elaborare i messaggi i modo da adattarli alle caratteristiche dei dispositivi stessi.
Non è più sufficiente utilizzare il nome per indirizzare i messaggi: molto spesso non si conosce il destinatario ma si ha comunque la necessità di scambiare informazioni con esso. Si rende quindi necessario utilizzare informazioni basate sul contesto operativo in vui si sta operando, come la posizione o le proprietà degli utenti e dei dispositivi che si trovano nella località.

In questo nuovo scenario risulta quindi importante il concetto di *comunicazione* tra membri dello stesso *gruppo* di lavoro creato in base al *contesto* in cui ci si trova a operare.

3.1 Modello di gestione dei Gruppi

AGAPE è un framework creato per supportare lo sviluppo di applicazioni collaborative in ambienti MANET. La collaborazione è basata sulla metafora di **gruppo** Ogni gruppo è rappresentato mediate un Group Identifier (GID) e un profilo che ne specifica interessi e obiettivi che si assume siano condivisi dagli utenti che ne fanno parte. I membri di un gruppo sono caratterizzati da un Personal Identifier (PID) e da un profilo che rappresenta le caratteristiche dell'utente e del dispositivo che esso utilizza. Sia i GID che i PID sono statisticamente univoci all'interno della rete.
I membri che faranno parte del gruppo non possono essere determinati a priori in quanto essi posso unirsi o lasciare il gruppo dinamicamente in base alle esigenze specifiche dell'applicazione. La mobilità degli utenti potrebbe creare partizioni e fusioni dei gruppi stessi e della rete, che causano fasi transitorie nella collaborazione fra utenti nuovi e precedentemente sconosciuti.
Il concetto di località è centrale in AGAPE per gestire e organizzare in gruppi.
Una località è definita come un insieme di entità di AGAPE i cui dispositivi sono connessi tramite un percorso di routing ha distanza minore di un determinato valore di soglia. Questo valore è definito in network hops e determina la massima dimensione che una località può assumere. Il numero di hop utilizzato per determinare il raggio della località può essere variato in base allo scenario applicativo.
Esistono due diverse tipologie di entità che possono operare all'interno del modello proposto da AGAPE:

- **ME**, Managed Entity: rappresentano quei membri del gruppo che utilizzano i servizi messia a disposizione dal supporto per collaborare;
- **LME**, Locality Manager Entity: rappresentano quei membri del gruppo che, oltre a collaborare, metto a disposizione servizi per il supporto alle operazioni necessarie per la gestione del gruppo stesso.

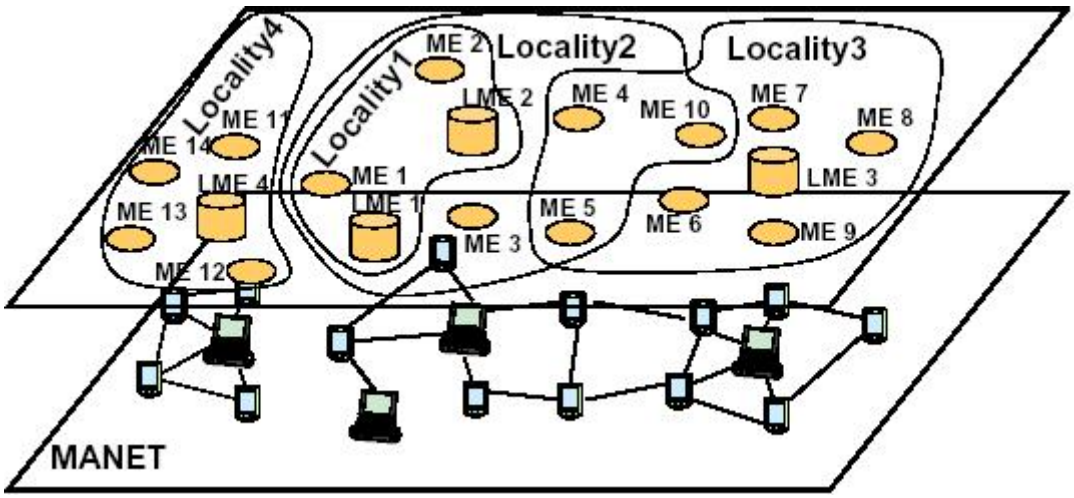

Figura 3.1: Concetto di località in AGAPE

Gli LME promuovono la creazione di nuovi gruppi e permettono alle altre entità presenti nella località di registrarsi presso i gruppi disponibili (consentono il join) e di mantenere e distribuire una lista dei membri del gruppo presenti nella località.
Ogni località deve contenere almeno un LME, che ne rappresenta il centro geografico. E' inoltre possibile una parziale sovrapposizione di due o più località dovuta alla presenza di più LME vicini fisicamente.
Gli ME possono muoversi liberamente e fare parte di più località contemporaneamente.
E' inoltre possibile che più LME facciano parte della medisima località; in questo caso solo uno svolgerà le funzioni proprie del Locality Manager Entity, mentre l'altro si limiterà a collaborare alla attività del gruppo.

3.1.1 Il Modello di Comunicazione

AGAPE mette a disposizione delle entità due differenti pattern di comunicazione: **context-based any-cast** e **context-based multi-cast**.
Il **context-based any-cast** permette la comunicazione asincrona inaffidabile point-to-point tra due utenti: quando un membro del gruppo ha bisogno di comunicare, uno e un solo membro della località viene selezionato in base al profilo di ricerca richiesto.
Il **context-based multi-cast** permette invece la comunicazione asincrona

inaffidabile point-to-multipoint: lo stesso messaggio viene inviato a tutti i membri il cui profilo corrisponde al profilo di ricerca specificato. Naturalmente questi due modelli proposti possono essere utilizzati come base per ottenere altri pattern più complessi con proprietà di sincronia e affidabilità. In entrambi i casi il destinatario del messaggio viene selezionato attraverso le sue caratteristiche e non in base al nome come avviene nei tradizionali modelli di comunicazione. Utilizzando questo sistema si ha il notevole vantaggio di non dover conoscere i membri di un gruppo ma permette di comunicare con membri le cui caratteristiche corrispondo a quelle da noi richieste.

3.2 L'architettura di AGAPE

AGAPE utilizza i servizi messi a disposizione dalla Java Virtual Machine (JVM) per intefracciarsi con il Sistema Operativo, la rete e l'hardware dei dispositivi su cui viene utilizzato.
AGAPE mette a disposizione diversi servizi per la gestione di gruppi in ambienti MANET.
I servizi sono organizzati in due strati logici (layer):

- **Group Management Layer**
- **Communication Layer**

Nella figura 3.2 viene indicata la struttura di agape appena descritta

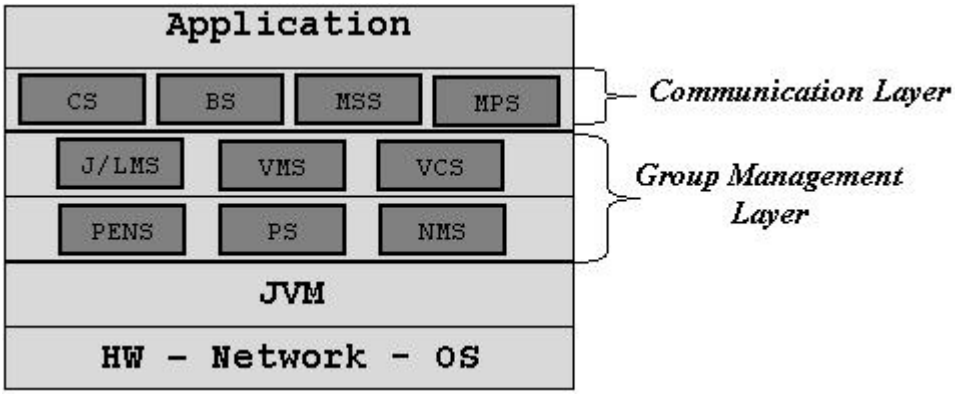

Figura 3.2: L'architettura di AGAPE

3.2.1 Group Management Layer

Il Group Management Layer mette a disposizione servizi per la creazione, la gestione e la distruzione di gruppi.
Esso è costituito dai seguenti servizi:

- **Network Management Service**: NMS;
- **Proximity Service**: PS;
- **Proximity Enabled Naming Service**: PENS;
- **Join/Leave Manager Service**: J/LMS;
- **View Manager Service**: VMS;
- **View Coordination Service**: VCS.

Network Management Service

L' NMS mette a disposizione degli altri servizi metodi per la comunicazione e lo scambio di messaggi.
Questo servizio implemente le primitive di comunicazione unicast, broadcast e multicast e garantisce un delivery efficiente dei messaggi in base alla specifica tecnologia di rete utlizzata.
Attraverso questo servizio le applicazioni, o gli altri servizi di AGAPE, possono selezionare l'interfaccia di rete da utilizzare e controllarne le proprietà, l'attività e la modialità operativa. Come già accennato l'NMS permette collegamenti punto-punto e punto-multipunto, (broadcast e multicast) di tipo connection-less.

Proximity Service

Il PS permette ai membri di AGAPE di notificare la loro presenza nella località.
Esso invia in broadcast a intervalli regolari un messaggio (beacon) contenente le seguenti informazioni:

- **GID-PID**: coppia di interi che identificano rispettivamente il gruppo a cui l'entità appartiene e il suo identificativo univoco;
- **RUOLO**: indica il ruolo che l'entità svolge nella località: LME o ME;

- **TTL**: il Time To Live indica il numero di hop per cui il messaggio deve essere propagato.

Ogni nodo che riceve il beacon vine quindi informato della disponibilità del nodo mittente e rispedisce il beacon con probabilità p(n) dopo aver decrementato il TTL.
In questo modo tutte le altre entità vengono avvertite della disponibilità on-line dell'entità mittente siano esse appartenenti allo stesso gruppo o meno.

Proximity Enabled Naming Service

Il PENS ha il compito di generare identificatori statisticamente univoci sia per i gruppi (GID) sia per i membri dei gruppi (PID).
Questo servizio costruisce e tiene aggiornata una tabella contenete indirizzo IP, GID, PID e stato di ogni entità di cui ha visibilità. Mette inoltre a disposizione primitive per conoscere i gruppi presenti nella località, tutte le entità e i loro ruoli. La tabella sopra citata viene costruita a partire dai beacon che vengono ricevuti dal PS; se una entità non invia più beacon la entry relativa ad essa viene eliminata dalla tabella.

Join/Leave Manager Service

Il J/LMS permette alle applicazioni che utilizzano AGAPE di creare gruppi, fare il join ad essi e lasciare un gruppo. Inoltre, attraverso il meccanismo di discover, permette di recuperare tutti i gruppi presenti nella località o solo quelli che rispondono ad un preciso profilo di ricerca.
Esistono due differenti versioni di questo servizio, una per LME e una per ME. E' stata effettuata tale distinzione in quanto gli LME sono le entità che sono in grado di gestire i gruppi al meglio in quanto dotati di maggiore autonomia e potenza di calcolo.
Gli LME possono quindi creare gruppi nuovi e gestire più gruppi contemporaneamente, mentre gli ME possono solo unirsi a gruppi già esistenti, anche più di uno anche se ne possono gestire solo uno per volta attraverso un switch esplicito. Inoltre solo gli LME rispondono alle richieste di join, leave o discovering dei profili.
La creazione del gruppo avviene chiedendo al PENS un GID casuale e un PID

per il primo membro del gruppo. Una volta creato il gruppo, viene chiesto al PS di diffondere la sua presenza nella località.
Quando invece si vuole effettuare il join ad un gruppo, il J/LMS recupera l'LME dal PENS e solo se il join va a buon fine il PS trasmette la coppia GID/PID.

View Manager Service

Il VMS si occupa di creare e distribuire le viste dei gruppi a tutti i membri. Ogni membro riceve una context-dependent view che contiene le informazioni su tutti i membri presenti nella località. Le informazioni consistono nel PID dell'utente, nel suo profilo e l'indirizzo IP. Quando un membro entra o lascia un gruppo o cambia il suo profilo, il VMS si occupa di notificare la nuova vista a tutti gli utenti interessati all'interno della località. Naturalmente il questo servizio collabora con il PENS per ottenere le notifiche dei nuovi arrivati o dei membri che sono andati via dalla località.

View Coordination Service

Il VCS è un servizio che permette ad AGAPE, e più precisamente al VMS, di decidere se inviare o meno le viste. Ad esempio se due LME fanno parte della stessa località, verrebbero disseminate le stesse viste da entrambi portando ad uno spreco di risorse come la banda e l'energia dei dispositivi.
Per ovviare a questo problema il VMS prima di inviare una vista chiede l'autorizzazione al VCS che confronta la vista che vuole essere inviata con l'ultima che ha ricevuto: se le entry sono diverse da il permesso al VMS di disseminare la sua vista; se le entry sono uguali il VCS sceglie un solo LME all'interno della località per l'invio delle viste. Il criterio di scelta è determinato in base al valore della batteria del dispositivo e nel caso di parità sceglie LME che presenta un PID maggiore.

3.2.2 Communication Layer

Il Communication Layer rappresenta lo stato superiore del framework AGAPE. Esso si basa sulle funzionalità offerte dai servizi del Group Management Layer.

Il Communication Layer mette a disposizione delle applicazioni che utilizzano AGAPE varie primitive per la comunicazione message-oriented e per l'adattamento a run - time dei messaggi e del loro scheduling.
Esso è costitutito dai seguenti servizi:

- **Communication Service**: CS;
- **Binding Service**: BS;
- **Message Presentation Manager Service**: MPMS;
- **Message Scheduler Service**: MSS.

3.2.3 Communication Service

IL CS supporta la comunicazione asincrona orientata ai messaggi e mette a disposizione delle applicasione delle primitice che implementano due modelli di comunicazione:

- Context-Based AnyCast;
- Context-Based MultiCast;

Per permettere una comunicazione un membro del gruppo deve fornire diverse informazioni: un profilo di ricerca **(Searching Profile)**, che specifica le sue preferenze di collaborazione, una strategia di binding **(early bindind o late binding)** e, nel caso dell'any-cast, un criterio **(Designation Criteria)** con cui scegliere un membro tra tutti quelli il cui profilo corrisponde al profilo di ricerca.
I dettagli implementativi di questo servizio verranno illustrati nei seguenti capitoli di questa tesi.

3.2.4 Bindind Service

Il BS ha il compito di gestire i binding tra le due parti di una comunicazione. Questo servizio supporta due differenti strategie di Binding:

- **Early Binding**: questa strategia determina un insieme di membri del gruppo (Target Member Set - TMS) il cui profilo corrisponde a quello di ricerca nel momento in cui il binding è creato; il binding cosi creato verrà utlizzato per l'intera sessione di comunicazione;

- **Late Binding**: questa strategia determina il Target Member Set dinamicamente ogni volta che un messaggio viene inviato.

Questo servizio, oltre a creare i binding attraverso il filtraggio della vista ottenuta dal VMS, si occupa di gestire una tabella **(Binding Table)** in cui vengono memorizzati i Binding richiesti dal Communication Service.
Anche per quanto riguarda questo servizio i dettagli implementativi verranno meglio esplicitati nel seguenti capitoli.

3.2.5 Message Presentation Manager Service

IL MPMS permette di scegliere e caricare dinamicamente differenti filtri per adattare il contenuto dei messaggi scambiati al contesto in cui si opera.
In particolare MPMS ottiene il messaggio dal MSS e usa il profilo dell'utente destinatario e del dispositivo fisico da lui utilizzato per applicare il filtro adatto tra quelli disponibili. Un esempio pratico potrebbe essere il ridimensionamento delle immagini ricevuto o la loro conversione in un formato più leggero per la loro spedizione su dispositivi con risorse limitate.

3.2.6 Message Schedueler Service

Il MSS assegna in modo dinamico una priorità per lo scambio dei messaggi. Tale priorità dipende dalle preferenze specifiche dell'applicazione. In particolare l'applicazione associa ad ogni profilo (utente/dispositivo/gruppo) un livello di priorità.
MSS costruisce un tabella **(Priority Table)** in cui vengono memorizzate queste priorità. Quando viene ricevuto un messaggio il servizio preleva dal VMS il profilo associato al mittente del messaggio e assegna a quel messaggio la priorità corrispondente cercandola nella tabella. Se non viene trovata nessuna corrispondenza viene assegnata una priorità di default. In questo modo è possibile inoltre specificare gli utenti non desiderati e, ad esempio, eliminare messaggi che non sono adatti alle attuali condizioni di lavoro.

Capitolo 4

Supporto per la Comunicazione Context-Aware in AGAPE

Il contesto delle rete ad-hoc è altamente variabile a causa dell'elevata mobilità dei nodi che costituiscono la rete stessa e alle frequenti connessioni e disconnessioni causate dalle limitate capacità energetiche dei dispositivi e dalla nauta delle comunicazioni wireless. Nel nuovo scenario creato dalle reti ad-hoc non è più possibile basare la comunicazione sulla conoscenza della posizione o sui nomi delle entità che ne fanno parte.
É chiaro che all'interno della rete Ad-Hoc è necessario un opportuno supporto al binding che ci consenta di creare binding e riqualificarli dinamicamente in accordo all' attuale situazione di rete. Il metodo di aggiornamento varia in base alla strategia di Binding scelta quando si è creato il binding stesso. AGAPE propone due differenti strategie: ***Early Binding*** e ***Late Binding***. Nel primo caso il binding non viene mai aggiornato una volta che è stato realizzato.
Nel caso di Late Binding il TMS è aggiornato alla disponibilità dei possibili partners presenti ogni volta che il binding viene utilizzato.
La scelta della strategia di binding è determinata da considerazioni legate al contesto applicativo; ad esempio, se si vogliono scambiare messaggi tipo chat con un utente verrà utilizzata la strategia Early in quanto molto probabilmente il membro con cui sto comunicando non andra via prima che la conversazione sia terminata.
Se invece la comunicazione è costituita da messaggi non frequenti, molto

probabilmente conviene utilizzare la strategia Late in quanto la rete in cui si opera è per sua natura dinamica.
Per quanto riguarda i modelli di comunicazione, AGAPE mette a disposizione due differenti pattern: la comunicazione ***any-cast*** e la comunicazione ***multi-cast***. Il primo pattern realizza la comunicazione punto-punto mentre il secondo rappresenta la comunicazione punto-multipunto.
In entrambi i casi il binding è ottenuto senza specificare un nome o un indirizzo, ma attraverso la specificazione delle caratteristiche che il destinatario deve avere. Come già spiegato in una rete ad-hoc molto spesso non si conoscono i membri del gruppo ma si desidera comunque comunicare e scambiare informazioni con essi.
Per permettere alle applicazioni di ottenere dei binding e di realizzare la successiva comunicazione sono stati sviluppati due nuovi servizi che si collocano nel Communication Layer di AGAPE: il **Communication Service** e il **BindingService**.
Il primo servizio è il punto d'accesso per le applicazioni che hanno necessità di inviare e ricevere messaggi. Il secondo è utilizzato dal Communication Service per ottenere i binding necessari alla comunicazione.
Di seguito verranno analizzate le principali funzionalità dei servizi sviluppati senza entrare eccessivamente nei dettagli implementativi, che verranno comunque discussi in seguito.

4.1 Ottenere il Binding

Quando un membro di un gruppo vuole stabilire una comunicazione il CS si coordina con il BS per creare un binding tra due membri del gruppo.
Il BS ottiene dal VMS la vista contenenti tutti i membri attualmente disponibili all'interno della località. Naturalmente questa vista dipende dal contesto in cui si sta operando nel momento in cui viene richiesto il binding.
Una volta ottenuta la vista il BS la filtra utilizzando come vincoli quelli contenuti nel profilo di ricerca indicato dal membro che vuole stabilire la comunicazione. Viene quindi creata un lista di membri che rispondono alle richieste imposte dal profilo di ricerca. Questo insieme, detto TMS, viene associato al membro del gruppo con una Binding Table che tiene traccia di

tutte le comunicazioni attive e restituisce un Handler che permette al mittente di stabilire un canale di comunicazione con il destinatario e di spedire il messaggio.
Il procedimento appena descritto è rappresentato in figura 4.1:

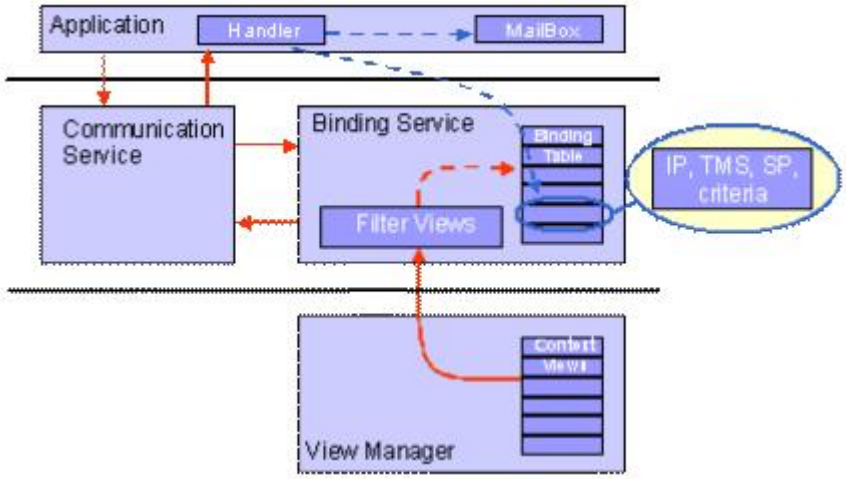

Figura 4.1: Creazione di un Handler e Binding Table

Ogni Handler rappresenta una entry nella Binding Table (rappresentata in figura 4.2), e ogni entry nella Binding Table rappresenta uno specifico binding ed include differenti campi:

SP	TMS	Criterio	BS	GID

Figura 4.2: Struttura della Binding Table

- Il profilo di ricerca (SearchProfile);
- L'insieme dei membri il cui profilo rispetta i vincoli (TMS;
- L'indirizzo IP di ogni membro presente nel TMS;

- La strategia di Binding adottata;
- Il criterio con cui il destinatario viene scelto (solo nel caso di context-based anycast).

Nella figura 4.3 vengono mostrate, attraverso un diagramma di sequenza, tutte le operazioni necessarie affinche un'applicazione possa creare un Binding, ottenere un Handler e spedire il messaggio.

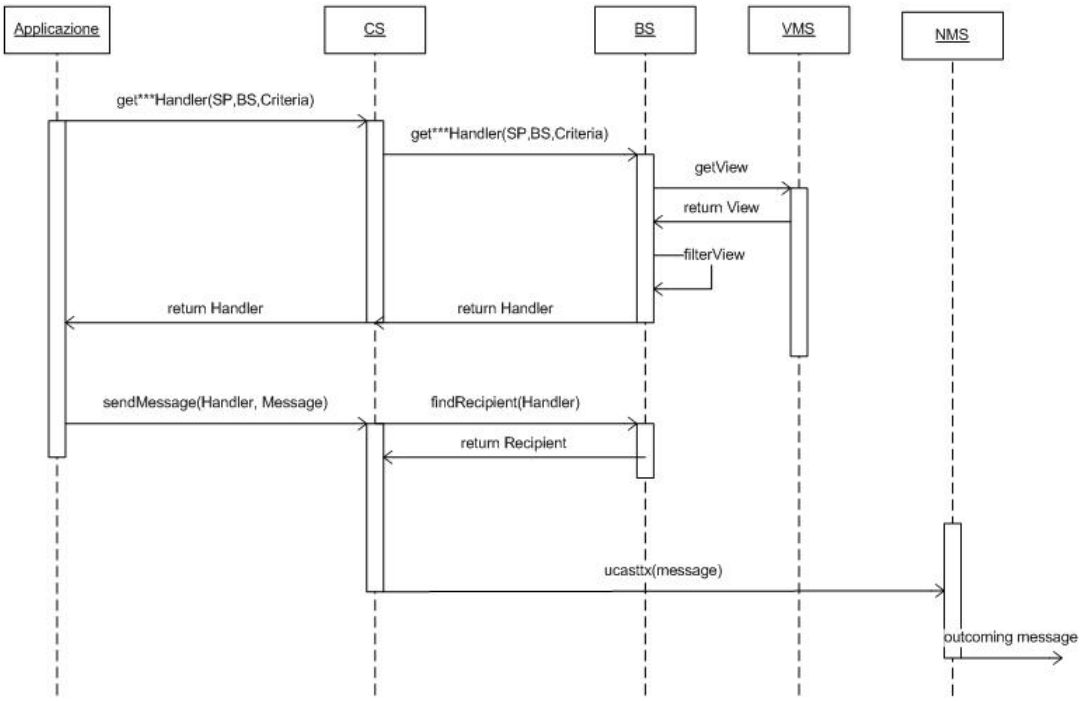

Figura 4.3: Diagramma di Sequenza per la creazione di un binding l'invio di un messaggio

1. L'applicazione chiede al CService un handler di tipo AnyCast con strategia early binding indicando un profilo di ricerca e il criterio di selezione
2. Il CS richiede il binding al BS che controlla nella BindingTable se esiste già una Entry; se cosi fosse utilizzerebbe il TMS presente nella tabella per selezionare un membro attraverso il criterio. Se non trova una entry con le caratteristiche richieste ne crea una nuova dopo aver ricevuto dal VMS la vista attuale e averne filtrato i membri. Una volta ottenuto il TMS il BS crea un handler che il CS può restituire all'applicazione;

3. una volta ottenuto il riferimento l'applicazione utilizza la primitiva send del CS per inviare il messaggio al membro;
4. il CS ottiene il destinatario del messaggio e invia il messaggio utilizzando la trasmissione unicast messa a disposizione dal NMS.

In caso di MultiCast il metodo send del CS manda lo stesso messaggio in unicast a tutti i membri presenti nel MuliCastHandler.

4.2 Inviare il messaggio

Una volta ottenuto il binding, l'applicazione può utilizzare le primitive messa a disposizione dal Communication Service per inviare il messaggio al destinatario prescelto.
É da notare che nel caso di comunicazione context-based any-cast il messaggio viene spedito ad uno e uno solo dei membri presenti nel TMS, in accordo con il SearchProfile, della strategia di Binding e del criterio prescelto. Nel caso di comunicazione di tipo context-based multi-cast il messaggio è consegnato a tutti i membri presenti nel TMS.

4.3 Ricezione di un messaggio

Il Communication Service è un componente attivo di AGAPE.
Esso è costantemente in ascolto su una determinata porta e alla ricezione di un messaggio crea un thread che si occupa di gestire il nuovo messaggio.
Nella figura 4.4 è rappresentata la sequenza di operazioni necessarie per ricevere un nuovo messaggio.

4.4 Communication Service

Il Communication Service (CS) fa parte del Communication Layer del framework AGAPE.
Esso mette a disposizione dell'utente le primitive per la comunicazione tra i membri dei gruppi e si occupa di notificare i messaggi che vengono ricevuti.

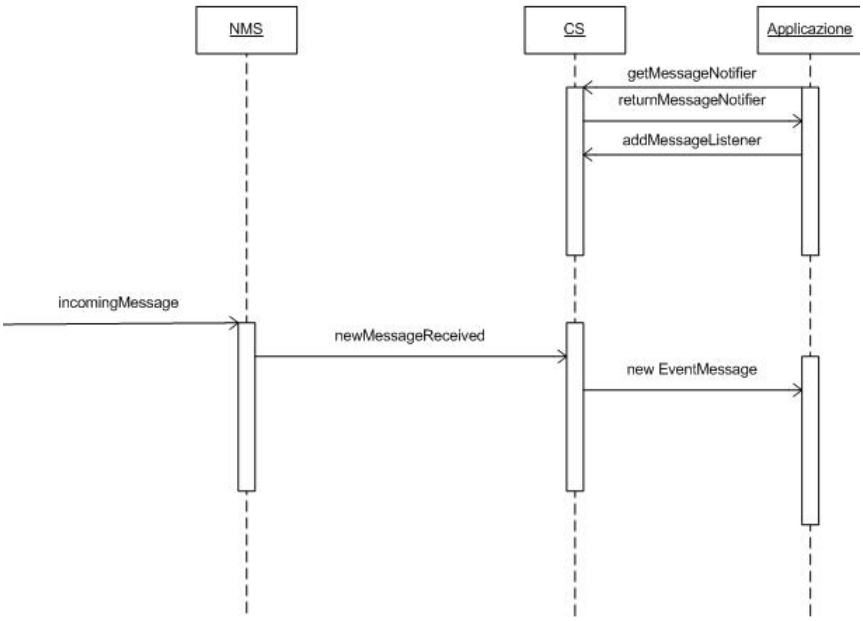

Figura 4.4: Diagramma di Sequenza per la ricezione di un messaggio

Per far ciò il CS si coordina con il Binding Service al fine di ottenere il binding, e con il NetManagerService per trasmettere il messaggio.
Le funzioni principali messe a disposizione dal CS sono:

- **Ottenere un riferimento**: permette di ottenere l' handler che verrà utilizzato per comunicare;
- **Inviare un messaggio**: l'applicazione una volta ottenuto l'handler può utilizzare le primitive messe a disposizione dal servizio per inviare messaggi;
- **Ricezione dei messaggi**: il servizio resta in ascolto su una determinata porta in modo da ricevere i messaggi inviati dagli altri membri del gruppo.

Abbiamo già visto che nel framework sono presenti due diverse entità: LME e ME.
Nel caso del Communication Service non ci sono differenze rilevanti; infatti, utilizzando il Binding Service, l'unica differenza è dovuta alla presenza del parametro che rappresenta il gruppo in cui si vuole cercare il membro con cui comunicare.

4.4.1 Diagramma statico delle classi

Come già detto nel paragrafo precedente, il Communication Service oltre a esporre dei metodi per permettere alle applicazioni di creare un handler e spedire un messaggio ha il compito di ricevere i messaggi provenienti da altre entità e a notificare tale messaggi alle applicazioni.Per questo motivo il servizio è un servizio attivo che deve restare in ascolto su una determinata porta: far ciò si è deciso di implementare il servizio in modo da estendere la classe ActiveComponent che è un AgapeComponent attivo come rappresentato in figura 4.5:

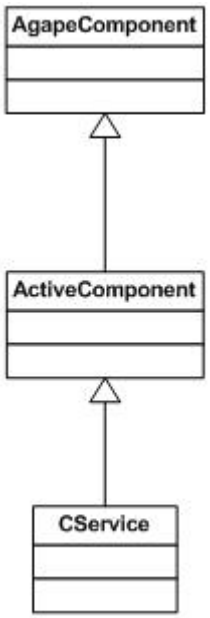

Figura 4.5: Tassonomia del Communication Service

Estendendo tale classe il Communication service potrà essere fatto partire attraverso il metodo ***start()*** e fermato con ***stop*** e il ***task*** si occuperà do attendere e notificare i messaggi in ingresso.
Le del contenute nel package del CS sono:

- **CommunicationService**: rappresenta il punto d'accesso alle funzionalita del CS;
- **InputThread**: è il thread che si occupa della gestione dei messaggi ricevuti;

- **MessageNotifier**: rappresenta la classe che si occupa di notificare i messaggi ricevuti alla applicazioni e gestisce la registrazione delle stesse;
- **MessageListener**: estende EventListener; è l'interfaccia che gli ascoltatori delle applicazione che intendono registrarsi per la ricezione dei messaggi in ingresso devono implementare;
- **MessageEvent**: estende EventObject e rappresenta l'evento che viene lanciato quando un nuovo messaggio viene ricevuto;
- **AgapeMessage**: è un oggetto che implementa *Serializable* e rappresenta il messaggio che le applicazioni si scambiano.

In figura 4.6 è rappresentato il diagramma UML delle classi utilizzate per implementare il Communication Service:

CService

La classe CService si occupa della ricezione dei messaggi e della loro notifica alle applicazioni e permette a quest'ultime di spedire a loro volta messaggi. I principali metodi di questa classe sono:

- ***getCBAnyCastHandler***: attraverso questo metodo è possibile ottenere un handler di tipo AnyCast da utilizzare per inviare il messaggio al destinatario; come si nota dal diagramma UML in figura 4.6 questo metodo ha tre o quattro parametri:
  - *SearchProfile*: di tipo ProfileImpl; rappresenta il profilo di ricerca;
  - *BindingStrategy*: di tipo String; rappresenta il modello di comunicazione che si vuole adottare, può assumere due valori: EARLY-BINDING o LATEBINDING;
  - *Criterio*: di tipo String; rappresenta il criterio di scelta tra tutti i membri il cui profilo rispetta i vincoli.
  - *GID*: di tipo int; indica il gruppo in cui effettuare la ricerca ma è presente solo negli LME
- ***getCBMultiCastHandler***: attraverso questo metodo è possibile ottenere un handler di tipo MultiCast da utilizzare per inviare un messaggio al destinatario; come si nota dal diagramma UML, anche questo metodo ha gli stessi parametri di quello precedente ad eccezione del *Criterio*; infatti in caso di MultiCast il messaggio verrà inviato a tutti i membri il cui profilo rispetta i vincoli espressi nel SearchProfile.

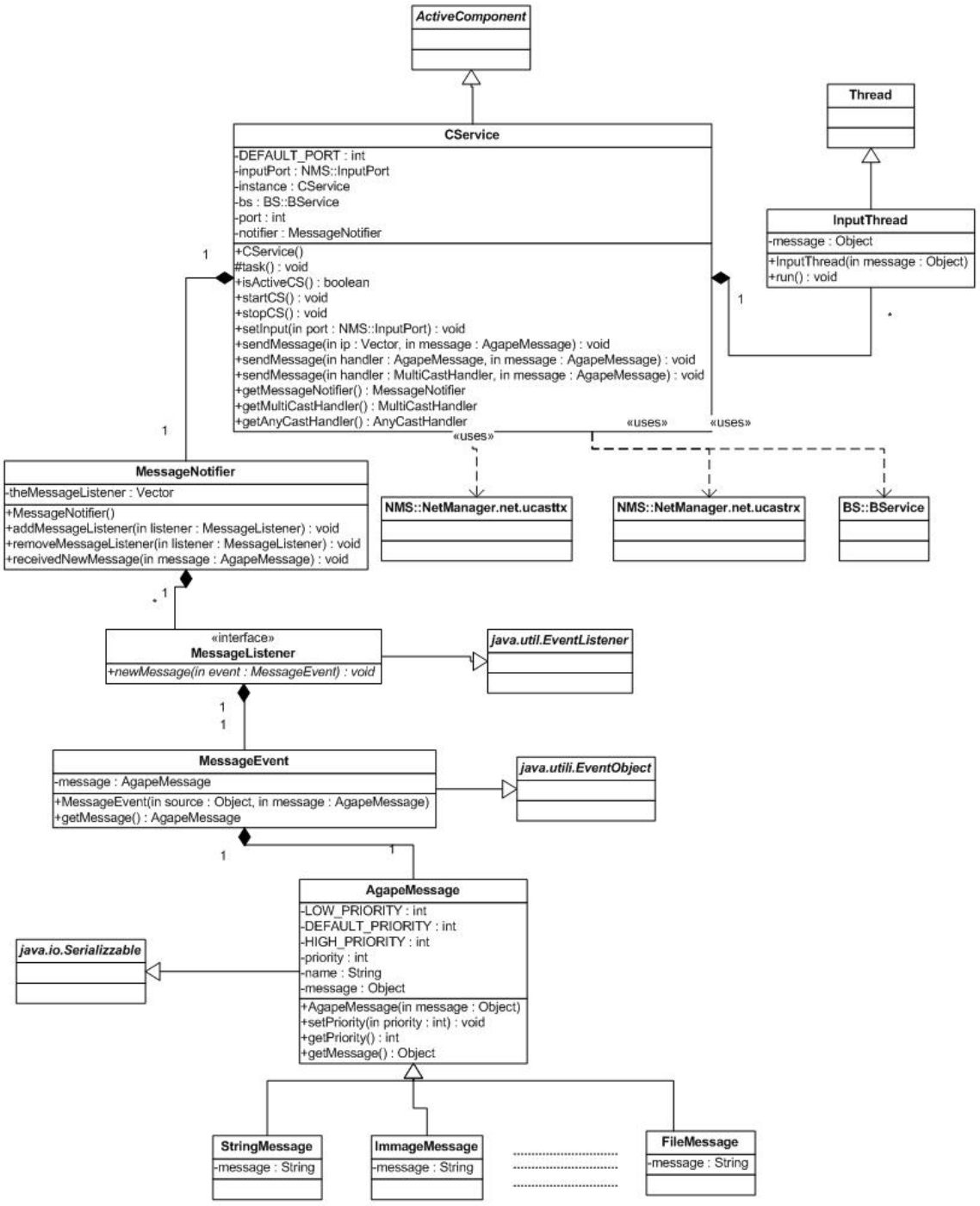

Figura 4.6: Diagramma statico delle classi del Communication Service

- ***SendMessage***: presente in diverse forme; permette di inviare un messaggio (di tipo AgapeMessage) ad un handler (sia AnyCast che MultiCast);
- ***task***: non è un metodo che può essere invocato direttamente ma parte automaticamente quando il servizio viene avviato. Questo metodo attende i messaggi e, quando arrivano, crea un **InputThread** che si occuperà di processare il messaggio inn modo opportuno.

InputThread

Questa classe estende Thread è viene creata dal *task* del CService quando arriva un nuovo messaggio. Il CService crea un nuovo InputThread passando anche il messaggio ricevuto e si rimette subito in ascolto.
Il nuovo thread creato si occupa di controllare che il messaggio ricevuto sia un AgapeMessage e, se è così, lo comunica al MessageNotifier che si occuperà di informare tutte le applicazioni registrate. Una volta fatto questo l'oggetto ha esaurito la sua funzione quindi in Thread muore.

MessageNotifier

Il MessageNotifier è un oggetto creato nel momento in cui il Communication Service viene attivato.
Le applicazioni che utilizzano il CS ottengo da esso un riferimento al MessageNotifier e, se sono interessate alla ricezione di messaggi, registrano il loro listener attraverso il metodo ***addMessageListener***; dualmente possono rimuovere il loro listener dal MessageNotifier attraverso il metodo ***removeMessageListener***.
Quando l' InputThread riceve un AgapeMessage invoca il metodo ***receivedNewMessage*** del MessageNotifier che genera un MessageEvent e si occupa di notificarlo a tutti il Listener registrati.

AgapeMessage

Questa classe è stata creata in modo da poter aggiungere al semplice Object alcune funzionalità che potrebbe essere utilizzate in futuro da altri servizi

come il MPS o, nel caso della priorità, dal MPSS. AgapeMessage implementa l'interfaccia Serializable in modo da poter essere scritto e letto su o da un stream come le socket. Inoltre mette a disposizione dei livelli di priorità, rappresentati da un intero e una proprieta *name* che può essere utilizzata dalle applicazioni per indicare il nome del messaggio senza creare necessariamente delle sottoclassi di AgapeMessage.

4.5 Binding Service

Anche il Binding Service (BS) è un servizio del framework AGAPE. Esso fa parte del Group Communication Layer e sfrutta i servizi messi a disposizione dal Group Management Layer già sviluppati in altre tesi, in particolare del VMS.
Tale servizio permette all'applicazione di ottenere dei binding con altri utenti del gruppo. In pratica in BS implementa il gestore della Binding Table e mette a disposizione i metodi per crearla, modificarla e interagire con essa.
Le principali funzioni messe a disposizione da questo servizio sono:

- **Creazione della tabella**: permette alla applicazione che lo utilizza di creare una Binding Table le cui caratteristiche saranno viste in seguito;
- **Gestione della tabella**: permette di aggiungere entry alla tabella e di eliminarle;
- **Ottenere un riferimento AnyCast:** permette di ottenere un riferimento ad un solo membro che corrisponde ad un determinato profilo di ricerca;
- **Ottenere un riferimento MultiCast**:permette di ottenere un riferimento a tutti i membri il cui profilo combacia con quello di ricerca.

Nel framework sono presenti due diverse entità, LME e ME, che differiscono per capacità computazionali ed energetiche.
In particolare l'entità di tipo LME è in grado di gestire più gruppi contemporaneamente. Come già detto, questo servizio è strettamente correlato con il View Manager Service del Group Management Layer. La versione per LME, oltre ad eseguire tutte le funzioni della versione per ME, permette anche di memorizzare il GID del gruppo in cui si è effettuata la ricerca. In questo

modo si sfruttano tutte le potenzialità messe a disposizione dalla versione per LME del VMS che permette di gestire più gruppi contemporaneamente. Oltre ad utilizzare il VMS per ottenere i profili dei membri presenti al momento della richiesta di binding, il Binding Service utilizza anche le classi relative alla creazione e al confronto dei profili, in particolare sfruttando il Profile Matcher.

4.5.1 Diagrammi statico delle classi

Il Binding Service implementa il gestore della Binding Table, mette cioè a disposizione delle applicazioni o dei servizi che lo utilizzano dei metodi per la creazione e la gestione della tabella, per l'inserimento e l'eliminazione delle entry e per la sua consultazione.
Per questo motivo si è scelto di implementarlo come un oggetto che contiene la Binding Table, sulla quale è possibile operare attraverso i metodi che esso espone.
Le classi contenuto nel package del BS sono:

- **BindingService**: rappresenta il punto d'accesso alle funzionalita del BS;
- **BindingTable**: implementa la tabella in cui vengono conservati i dati;
- **BTEntry**: rappresenta una entry della tabella;
- **TMS**: Target Member Set, è un campo del BTEntry e l'insieme dei membri il cui profilo corrisponde a quello di ricerca;
- **AnyCastHandler**: realizza un handler di tipo anycast;
- **MultiCastHandler**: realizza un handler di tipo multicast;

Oltre a queste classi è stata sviluppata una classe di appoggio per il filtraggio dei profili: **FilterUtility**.
Questa classe permette di ottenere un inseme di vincoli (Constraint) a partire da un profilo di ricerca (SearchProfile); in questo modo è possibile utilizzare i metodi messi a disposizione dalla classe ProfileMatcher contenuta nel package com.sun.ccpp.

In figura 4.7 è riportato il diagramma UML delle classi utilizzate per implementare tale servizio.

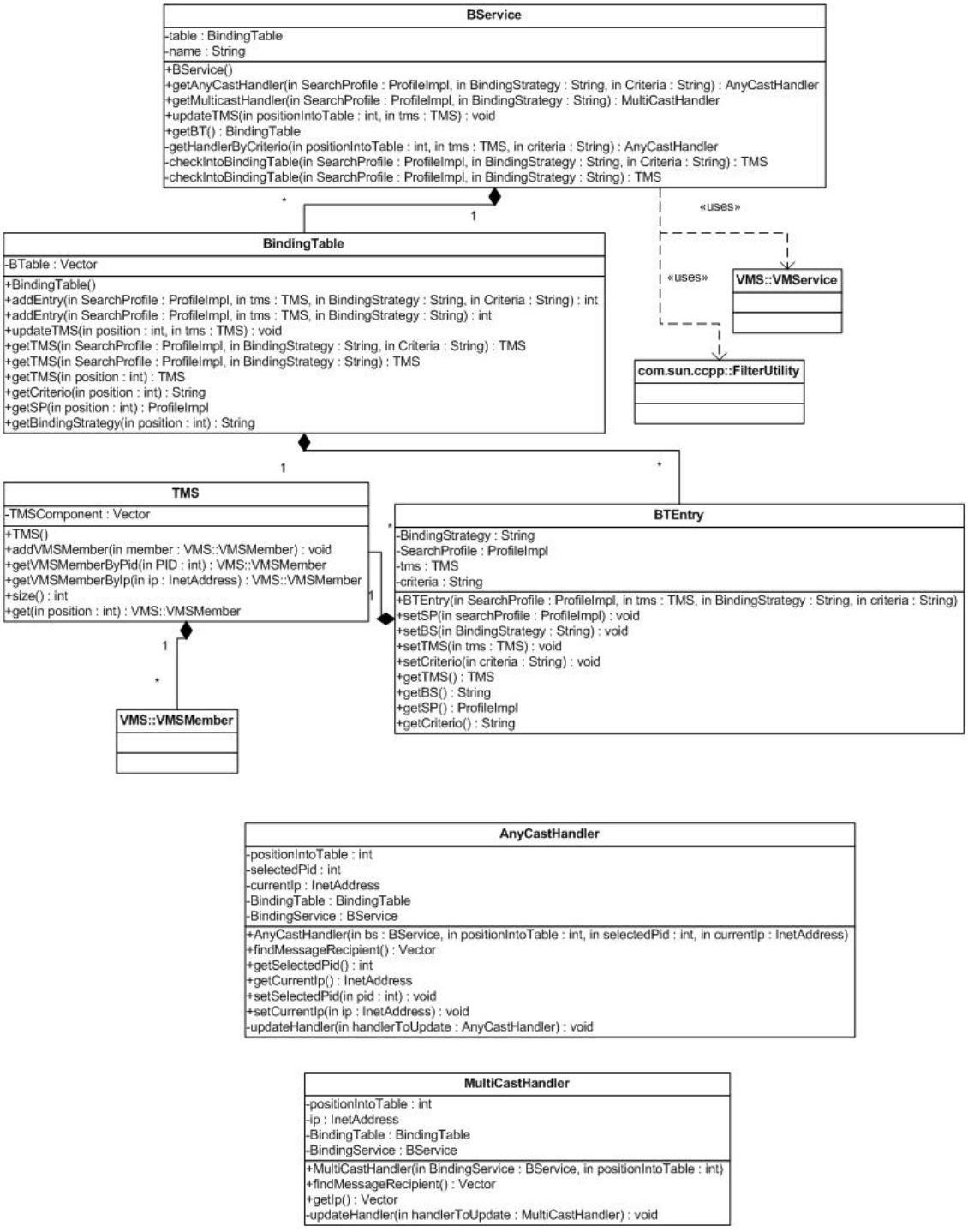

Figura 4.7: Diagramma statico delle classi del Binding Service

BindingService

Come già detto questa classe rappresenta il punto di ingresso a tutte le funzionalità di questo package.
Le principali primitive messe a disposizione dei clienti sono:

- **getAnyCastHandler**: questo metodo permette di ottenere un handler di tipo AnyCast; come si nota dal diagramma UML, questo metodo ha tre o quattro parametri:

  - *SearchProfile*: è di tipo ProfileImpl e rappresenta il profilo di ricerca con cui confrontare i profili ottenuti dal VMS per ottenere l'handler per la comunicazione
  - *BindingStrategy*: è di tipo String e rappresenta la strategia di Binding che si vuole utilizzare; può assumere due valori: EARLYBINDING o LATEBINDING.
  - *Criteria*: è di tipo String e indica al servizio il criterio con cui selezionare il destinatario del messaggio tra tutti quelli il cui profilo corrisponde a quello di ricerca; per ora sono implementati due criteri: FIRST o LAST ma possono essere facilmente essere implementati altri metodi in base al contesto operativo, come ad esempio il valore della batteria o la vicinanza;
  - *GID*: questo parametro indica il gruppo in cui effettuare la ricerca ma è presente solo negli LME.

- **getMultiCastHandler**: questo metodo permette di ottenere un handler di tipo MultiCast; dal diagramma UML si nota che, non dovendo specificare un criterio per la selezione di un membro, i parametri in questo metodo sono solo due:

  - *SearchProfile*: è di tipo ProfileImpl e rappresenta il profilo di ricerca con cui confrontare i profili ottenuti dal VMS per ottenere l'handler per la comunicazione
  - *BindingStrategy*: è di tipo String e rappresenta la strategia di Binding che si vuole utilizzare; può assumere due valori: EARLYBINDING o LATEBINDING.

Filtraggio delle viste

Per ottenere l'insieme dei membri il cui profilo rispetta i vincoli imposti dal profilo di ricerca viene utilizzato un componente già sviluppato in un' altra tesi: il **ProfileMatcher**.
Questa classe permette di confrontare un profilo con un vettore di Constraint secondo un determinato algoritmo di matching. Come primo passo si devono estrarre i vincoli dal profilo di ricerca; per fare cio si utilizza il metodo sviluppato appositamente ***getConstraintsByProfile*** della classe **FilterUtility**. Dopo aver fatto questo si crea un oggetto della sottoclasse opportuna di ***Constraint*** e alla fine si restituisce un vettore contenete tutti i vincoli specificati nel profilo di ricerca.
Una volta ottenuto questo vettore viene passato al ProfileMatcher insieme a un vettore contenente tutti i profili dei membri presenti nella vista. In questo modo si otterranno solo i membri che rispettano i vincoli secondo l'algoritmo di matching specificato.
Gli algoritmi implementati sono tre;

- **PerfectMatch**: i vincoli devo essere presenti nel profilo e devono essere tutti verificati; viene inoltre verificata anche la dimensione del profilo.
- **CompleteMatch**: meno rigoroso del precedente, controlla che i vincoli passati siano tutti verificati se presenti nel profilo;
- **PartialMatch**: controlla che sia verificata almeno una condizione.

Per questa versione del BindingService si è scelto di utilizzare l'algoritmo **CompleteMatch** ma questa scelta è facilmente modificabile in base alle necessità dell'applicazione.
Nelle figura 4.8 è rappresentato l'algoritmo in modo schematico per specificarne meglio il suo funzionamento.

BindingTable

In seguito alla richiesta di un handler fatta da una applicazione al Binding Service viene inserita una entry nella BindingTable.
Questa classe è quindi la struttura dati alla base del servizio ed può essere schematizzata come in figura4.2.

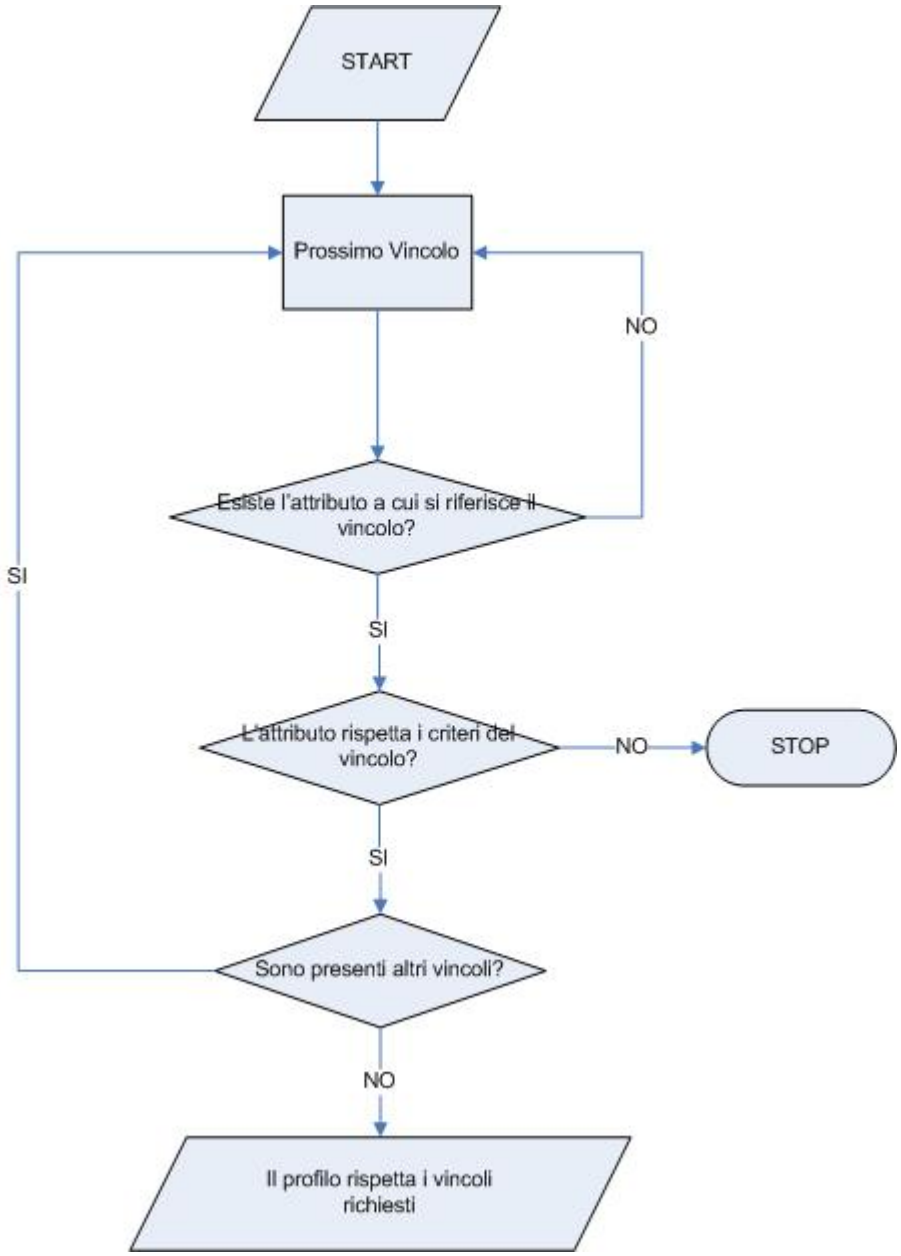

Figura 4.8: Algoritmo CompleteMatch

Ogni riga della tabella rappresenta il risultato di una richiesta di binding da parte dell'applicazione. E' quindi possibile conservare l'insieme dei membri che soddisfano un determinato pattern di ricerca e, ad esempio riutilizzarlo in caso di Early Binding.

BTEntry

Come già sottolineato precedentemente questa classe implementa una riga delle Binding Table. I campi della entry sono realizzate utilizzando le seguenti proprietà:

- **SP**, *SearchProfile*: è di tipo ProfiliImpl e rappresenta il profilo con cui si vuole stabilire una connessione;
- **TMS**, *Target Member Set*: è realizzato tramite la classe TMS e rappresenta l'insieme di tutti i membri che corrispondono al profilo richiesto;
- **Criterio**: è di tipo String, indica quale membro deve essere prescelto all'interno del TMS come destinatario della comunicazione; in caso di richiesta di multicast è null, quindi può essere anche utlizzato per distinguere tra le due richieste.
- **BS**, *Binding Strategy*: è di tipo String, può assumere due valori (EarlyBinding oppure LateBinding);
- **GID**, *Group IDentifier*: identifica il gruppo in cui si è fatta la ricerca; questo campo è presente solo nella versione del servizio per LME, in quanto solo queste entità hanno la possibilità di gestire più gruppi contemporaneamente.

TMS

Una volta recuperati i membri dal VMS e dopo averli filtrati viene creato il TMS attraverso la classe omonima. L'oggetto cosi ottenuto è contenuto in una riga della BindingTable e contiene oggetti di tipo VMSMember da cui è possibile estrarre

- Profilo del membro;
- Indirizzo IP del membro;
- PID del membro.

Mette a disposizione i seguenti metodi per inserire nuovi membri e per recuperarli:

- **addVMSMember**: permette di aggiungere un membro al TMS passandolo come argomento;
- **getVMSMemberByPid**: restituisce il membro corrispondete al PID passato come parametro; se non c'è nessun membro con il PID specificato restituisce null;
- **getVMSMemberByIp**: restituisce il membro il cui indirizzo IP corrisponde con quello passato come parametro, se non c'è nessun membro con il PID specificato restituisce null;
- **get**: restituisce il membro che si trova nella posizione corrispondente a quella passata come parametro.

Handler

Un handler rappresenta il riferimento al membro o ai membri che sono stati ottenuti dopo il filtraggio della vista attuale del VMS. In particolare esistono due tipi di handler: il MultiCastHandler e l' AnyCastHandler. Entrambi contengo le seguenti informazioni:

- **positionIntoTable**: rappresenta la posizione all'interno della BindingTable;
- **BindingTable**: il riferimento alla Binding Table di appartenenza;
- **BindingService**: il riferimento alla servizio di Binding;

Attraverso il metodo ***findMessageRecipient*** è possibile recuperare l'indirizzo ip del destinatario quando viene inviato il messaggio; in caso di comunicazione anycast verrà restituito un solo indirizzo ip, mentre in caso di multicast verrà restituita una lista di ip a cui inviare lo stesso messaggio.
Quando viene invocato tale metodo si controlla se la strategia di binding, contenuta nella tabella, è di tipo Early o Late; in quest'ultimo caso viene aggiornato il TMS e creato il Binding dinamicamente.

Capitolo 5

Caso di Studio: AGAPEmergency

Per verificare il corretto funzionamento dei nuovi servizi sviluppati, si è realizzato il prototipo di un applicazione collaborativa per provvedere al primo soccorso in caso di emergenza sanitaria.
L'applicazione in oggetto permette all'utente che ha un malore di lanciare l'allarme e di creare un nuovo gruppo di aiuto. Gli utenti che sono nelle vicinanze ricevono la richiesta di aiuto e possono decidere di intervenire. Sarà quindi possibile fornire aiuto immediato anche senza la presenza di un infrastruttura.
L'applicazione sfruttando AGAPE, e in particolare i nuovi servizi del Communication Layer, permette agli utenti presenti nella località di coordinarsi al fine di eseguire la serie di operazioni necessarie alla gestione dell'emergenza.
Il sistema AGAPE si adatta perfettamente alle esigenze di questo tipo di applicazione . Nonostante sia presente una infrastruttura di rete come GSM, GPRS o UMTS a disposizione degli utenti, AGAPE permette di creare una collaborazione tra utenti che non si conoscono tra di loro. In particolare il sistema consente, una volta creato il gruppo di emergenza, di individuare tutti i vicini e mantenere aggiornata la lista attraverso la diffusione delle context-view: l'applicazione può quindi utilizzare tale vista per stabilire la comunicazione context-based tra i membri del gruppo. Grazie alle primitive messe a disposizione dal Communication Service e dal Binding Service e pos-

sibile selezionare un membro o un insieme di membri il cui profilo rispetta i vincoli espressi dal Search Profile. In questo modo è possibile comunicare e chiedere aiuto a una certa tipologia di membri.
Nel caso specifico dell'applicazione realizzata sono previste due tipologie di utenti all'interno del gruppo, oltre al membro che richiede aiuto: il *paramedico* e l'*utente ordinario*.
Il prototipo dell'applicazione di primo soccorso utilizza AGAPE per creare dinamicamente un gruppo di supporto e di riunire in un unico gruppo il membro colto dal malore, i membri ordinari ed eventuali paramedici presenti nella località e permette la collaborazione tra tutti i membri al fine di prestare primo soccorso alla persona colta da malore.
Per identificare i membri del gruppo vengono utilizzati i profili dello standard CC/PP. Il profilo dell'utente, oltra ad descrivere le caratteristiche comuni a tutti gli utenti permette di specificare anche la loro qualifica: **Gente comune** o **Paramedico**. L'utente che chiede aiuto perche colto da malore potrebbe essere equipaggiato mediante dispositivi indossabili che permettono di monitorare i parametri vitali e che sono in grado di lanciare l'allarme in caso di malore. Gli altri utenti e i paramedici potrebbero essere equipaggiati con dispositivi mobili come Xybernauts MA-V o con palmari e laptop che permettono la connettività wireless attraverso il protocollo IEEE 802.11b.
Per i test dell'applicazione si sono utilizzati dei laptop con il sistema operativo Fedora Linux e la Java Virtual Machine J2SE 1.4 e con il motore di Speech IBM Via Voice a AGAPE LME. Per le entità di tipo ME si sono utilizzati dei PDA HP iPAQ 3970 con scheda wireless 802.11 e sistema operativo Familiar Linux (Blackdown jvm) o Windows CE (jeode jvm).
Per quanto riguarda l'entità LME, ossi il malato da soccorrere, si è sviluppata una interfaccia vocale utilizzando le Java Speech API.

5.1 Descrizione dell'utilizzo dell'applicazione

Non appena l'utente o i sensori distribuiti sul corpo rilevano un malore viene creato un gruppo di aiuto. LME si occupa quindi di promuovere dinamicamente tutte le informazioni riguardanti in gruppo appena creato. Il PS si occupa di notificare la sua presenza on line; il PENS genera un GID e un

PID; il VMS crea la prima vista contenente il GID/PID, l'indirizzo IP e il profilo del membro colpito da malore e comincia a disseminarla.
Ogni utente presente nella località che ha un dispositivo mobile acceso può rilevare la presenza del gruppo attraverso le funzionalità client-side del PENS. Una volta rilevato il gruppo può unirsi ad esso attraverso il J/LMS.
In questo modo la vista si aggiorna in base al contesto in cui ci si trova e tutti i membri del gruppo possono collaborare per risolvere la situazione.
Il dispositivo del membro colpito da malore dissemina a tutti i presenti una lista di cose da fare detta *todoList*. Questa lista è creata dinamicamente in base alla patologia dell'utente e contiene una serie di operazioni da eseguire per prestare soccorso.
Quando un utente riceve la todoList può eseguire una o più operazioni presenti nella lista oppure chiedere al dispositivo dell'ammalato di inviare un aiuto su come tale operazione va svolta.
Una volta eseguira l'operazione l'utente lo segnala al gruppo selezionando tale operazione nella todoList: il cambiamento verrà notificato a tutti gli utenti del gruppo.
É importante notare che oltre a svolgere le operazioni presenti nella todoList i membri del gruppo possono comunicare tra loro in modo da coordinarsi meglio.
É da notare che la gestione delle inconsistenze nella todoList, dovuta a partizioni di rete e successivi merge, vengono risolte utilizzando una semantica di tipo *at least one*: è meglio che due utenti chiamino il pronto intervento.

5.2 Dettagli Implementativi

Esistono due diverse implementazioni del prototipo realizzato: una per LME che permette di creare un gruppo e coordinarlo, l'altra per ME che permette agli utenti del gruppo di comunicare e gestire l'emergenza. Le differenze tra le due applicazioni verranno esplicitate nella prossima sezione ma è importante sottolineare che l'applicazione realizzata permette lo scambio di messaggi tra gli utenti appartenenti allo stesso gruppo. In particolare è necessario che l'utente che chiede aiuto possa distribuire al lista dei task e che gli utenti possano scambiarsi le modifiche effettuata alla lista stessa. Un altro impor-

tante messaggio da considerare è l'howtoMessage che contiene le istruzioni necessarie all'esecuzione del task.
La **todoList** viene diffusa attraverso un AgapeMessage. Essa non è altro che un contenitore di *todoItem* che rappresentano il task da eseguire. Nella figura 5.1 viene rappresentato il diagramma UML di un *todoListMessage*:

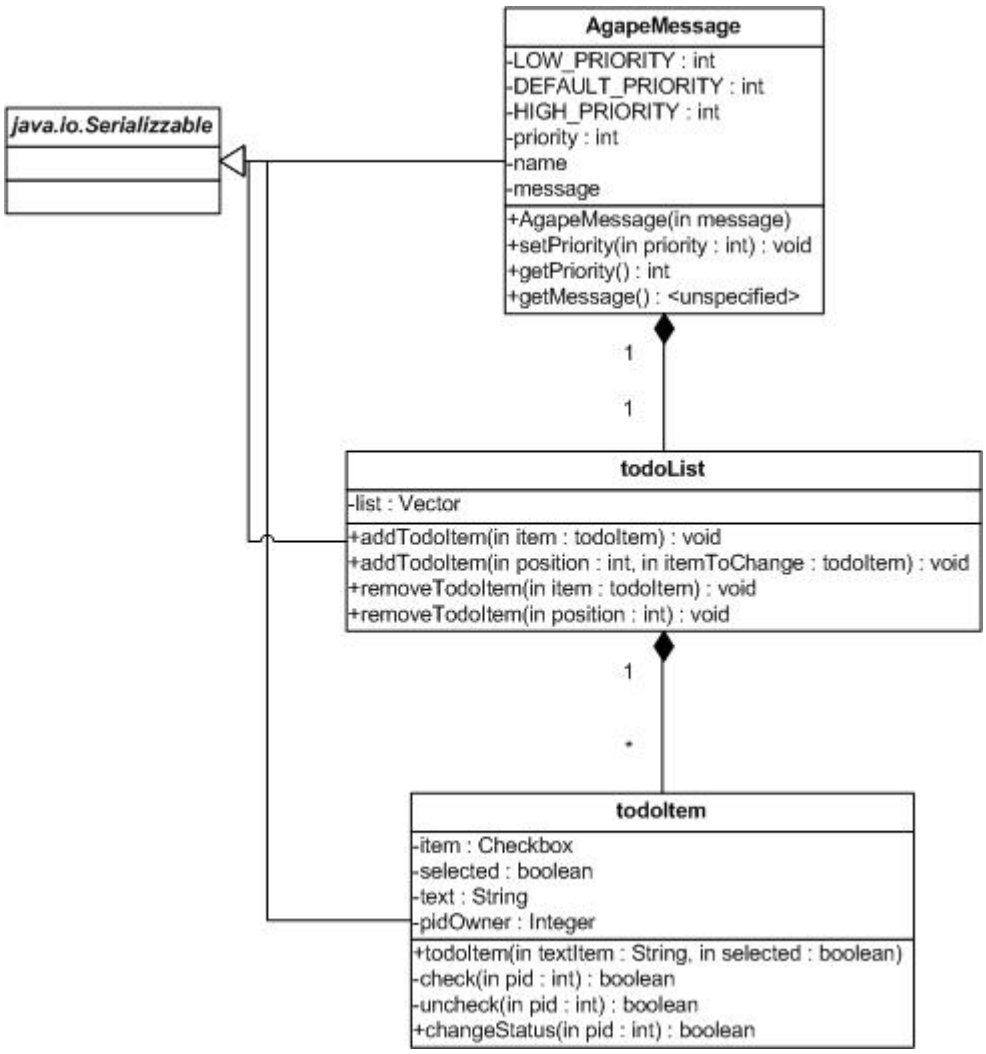

Figura 5.1: Diagramma UML di un todoListMessage

Anche le istruzioni vengono inviate nello stesso modo. L'informazione riguardante un singolo task è rappresentata da un Panel che è contenuto in un contenitore (howtoItem) che indica il task a cui si riferisce. Nella figura 5.2 vine rappresentato il diagramma UML di un *howtoMessage*:

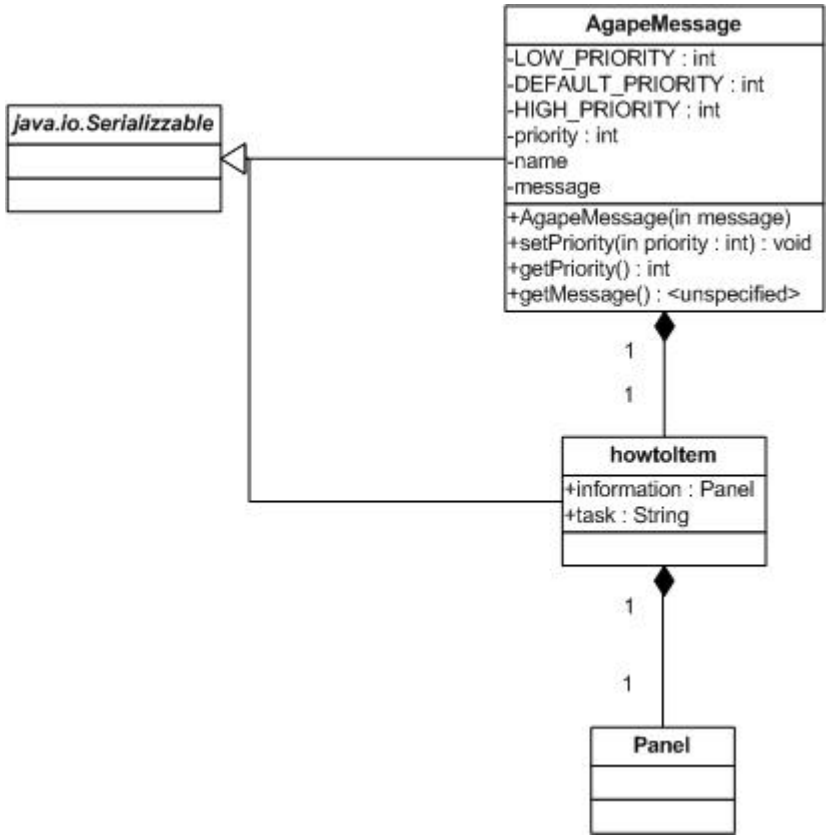

Figura 5.2: Diagramma UML di un howtoMessage

5.2.1 Versione LME

Il prototipo realizzato per l'utente che ha bisogno di richiedere aiuto sfrutta la versione LME di AGAPE.
Esso ha infatti bisogno di creare un gruppo in caso di malore e deve utilizzare l'interfaccia vocale per l'interazione con l'utente. Queste due caratteristiche presuppongono che l'applicazione sia eseguita su dispositivi che abbiano una capacità computazionale maggiore di quella necessaria agli ME. L'applicazione è progetta in modo da essere eseguita automaticamente all'avvio del dispositivo. All'avvio vengono avviati anche i servizi del framework agape e il sistema è pronto per lanciare l'allarme e creare il gruppo di aiuto.
Il profilo del gruppo e quello dell'utente, oltre a indicare tutte le informazioni anagrafiche, indicano anche attraverso un attributo il tipo di utente e il tipo di gruppo che si è creato. LME crea il gruppo di primo aiuto e il tipo di utente in questo dispositivo indica che si tratta di un *Anziano*.

Diagramma dei casi d'uso

La versione per LME di AGAPEmergency è il punto di partenza dell'intera rete. Solo da questa versione è possibile lanciare l'allarme, creare il gruppo e gestire il join dei membri e coordinare tutte le operazioni di soccorso.
Le principali funzioni del sistema sono:

- **Lanciare l'allarme**: agendo sul tasto ***HELP*** è possibile segnalare il malore;
- **Creare il gruppo di aiuto**; viene creato il gruppo di primo soccorso e l'utente diventa il primo membro del gruppo;
- **Coordinare la gestione delle emergenze**: distribuire la lista dei task, rispondere alle richieste di howto e gestire i conflitti;
- **Mettere fine all'emergenza**: una volta terminata l'emergenza è possibile lasciare il gruppo di aiuto attraverso il tasto ***EMERGENCY FINISHED***.

Una altra importante funzione di questa versione è rappresentata dal feedback vocale; l'applicazione infatti segnala attraverso interfaccia vocale tutti gli eventi che si susseguono nel corso dell'emergenza come l'ingresso nel gruppo di un nuovo soccorritore o l'esecuzione di un task o, semplicemente, il corretto funzionamento del sistema e la creazione del gruppo di aiuto.

Funzionamento dell'applicazione

Il prototipo per LME è costituito principalmente dall'interfaccia uomo - macchina e da una classe che permette di accedere a tutte le funzionalità del framework AGAPE.
L'interfaccia uomo-macchina è rappresentata, nel nostro caso, da un semplice pannello che permette chiedere aiuto e di dichiarare la fine dell'emergenza. Per quanto riguarda il feedback con l'utente viene utilizzata un interfaccia Text To Speech. Naturalmente il pannello per lanciare l'allarme potrebbe essere sostituito da sensori distribuiti sul corpo o sugli abiti per il controllo delle funzioni vitali che lanciano l'allarme non appena se ne presenta il bisogno.
Appena l'utente ha un malore chiede aiuto agendo sul tasto ***HELP*** (figura

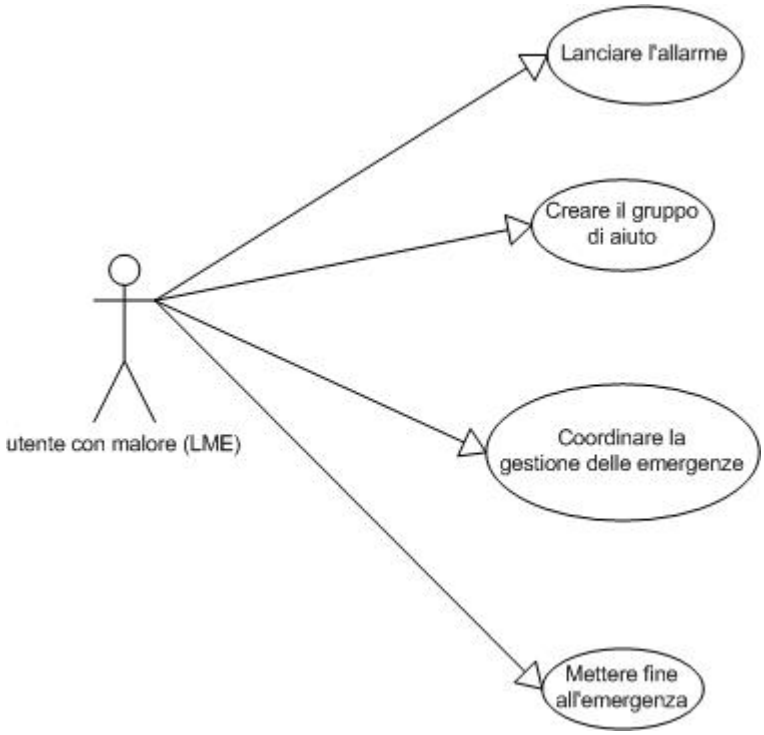

Figura 5.3: Casi d'uso per AGAPEmergency LME

5.4). Il sistema a questo punto cerca di creare il gruppo di aiuto e crea una lista di cose da fare a partire da un file di configurazione.
Le principali classi di questa versione sono le seguenti:

prototipoMain : è il punto di partenza dell'applicazione; si occupa del setup di agape e della corretta inizializzazione dell'interfaccia utente;

todoThread : è il thread che si occupa di inviare la lista dei task a tutti i membri del gruppo.

MyMessageListener : implementa l'interfaccia *MessageListener* del Communication Service e permette quindi la ricezione dei messaggi;

facade ; permette di accedere alle funzioni del framework.

prototipoMain

Questa classe contiene il main dell'applicazione ha tre compiti principali: inizializzare l'interfaccia vocale, avviare AGAPE e far partire l'interfaccia di AGAPEmergency.
Nel primo caso viene creato un thread che ha il compito di allocare le risorse

Figura 5.4: Richiesta d'aiuto e creazione del gruppo

necessarie al sistema di Text To Speech. Questo thread mette a disposizione dell'applicazione il metodo *say(String toSpeech)* e il metodo *deallocate* per liberare le risorse utilizzate dal motore vocale.
Per quanto riguarda il secondo compito viene eseguito il metodo **agapeStart()** che avvia tutti i servizi attivi di agape: **PENS**, **PS**, **VMS**, **J/LMS** e **CS**.
Infine viene creata il frame dell'applicazione che permette di chiedere aiuto.

todoThread

Quando viene richiesto aiuto viene creato un nuovo gruppo attraverso il J/LMS e il PS inizia a cercare nuove entità nella località. Quando le nuove entità rispondono alla richiesta di aiuto e si uniscono al gruppo il VMS inizia a disseminare le viste.
Il ruolo svolto dal todoThread è di distribuire a tutti i membri del gruppo la lista dei task da eseguire e le eventuali modifiche.

MyMessageListener

Questa classe implementa il listener del Communication Service e permette di registrarsi per la ricezione dei messaggi in ingresso.
Il listener della versione per LME supporta e riconosce tre tipo di messaggi:

textMessage : il semplice messaggio di testo;

updateItemMessage : indica la modifica da parte di un ME di un task della lista;

howtoMessage : rappresenta la richiesta di istruzioni per l'esecuzione di un task.

All'interno dell'applicazione i messaggi sono di tipo agape.cs.AgapeMessage (figura 5.5) e si differenziano dal campo **name** mentre il messaggio vero e proprio si trova nel campo **message**.

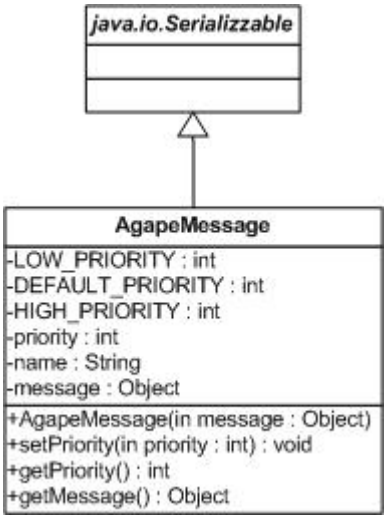

Figura 5.5: AgapeMessage utilizzato per l'invio dei messaggi

Quando viene ricevuto un textMessage viene utilizzata l'interfaccia vocale per comunicare il messaggio all'utente. Se viene ricevuto un howtoMessage viene cercata l'informazione richiesta all'interno di un hastable e restituita attraverso un howtoResponse. Infine per i messaggi di tipo updateItemMessage si controlla prima che il task non sia gia stato eseguito e si modifica la lista in modo che l'item risulti eseguito impostando come proprietario di tale item il pid del membro che lo ha eseguito. Se invece il task è già stato eseguito si controlla se il pid del task corrisponde con quello del membro

che ha inviato la richiesta e solo se questo avviene la lista viene modificata reimpostando il proprietario del task a null.

facade
Questa classe è realizzata mediante il pattern *facade*. Essa rappresenta un punto d'accesso comune per l'applicazione a tutti i servizi messi a disposizione da AGAPE.
I metodi utilizzati per la versione LME del prototipo sono:

createGroup utilizza il J/LMS per creare un nuovo gruppo;

notifyInputMessage permette di registrare il listener presso il CS per ricevere i messaggi in ingresso;

leaveGroup permette di lasciare il gruppo e mettere fine all'emergenza

sendMessageToAllMember permette di inviare i messaggi a tutti gli utenti del gruppo, come ad esempio la todoList.

5.2.2 Versione ME

Quando un utente riceve la richiesta d'aiuto può unirsi al gruppo di primo soccorso. Una volta effettuato il join riceve una lista di task da eseguire e, una volta eseguiti, deve essere in grado di comunicarlo al gruppo in modo che nessun altro esegua lo stesso task. L'applicazione deve inoltre permettere la richiesta e la consultazione delle istruzioni per eseguire i diversi compiti contenuti nella lista e infine deve permettere l'uso del Communication Service per lo scambio di messaggi fra i diversi membri del gruppo.
La versione per ME necessita di meno risorse e può quindi essere eseguita su dispositivi con limitata capacità computazionale come i PDA.
Anche in questa versione l'applicazione avvia tutti i servizi di AGAPE e cerca un gruppo fino a quando non viene rilevato. Una volta fatto il join a tale gruppo si è pronti a coollaborare con gli altri membri del gruppo. Quando viene fatto il join il tipo di utente specificato nel profilo può essere di due tipe: *odinary people* e *paramedico*.

Diagramma dei casi d'uso

Lo scopo principale della versione ME di AGAPEmergency e quello di coordinarsi con gli altri membri ME del gruppo per eseguire correttamente i task contenuti nella ***todoList***. Per far ciò si avvale del supporto alla comunicazione di gruppo messo a disposizione dal Communication Service.
Le principali funzioni del sistema sono:

- **Rilevare la presenza di un gruppo di primo soccorso**: attraverso un thread viene interrogato periodicamente il J/LMS per sapere se è presente il gruppo;
- **Unirsi al gruppo di aiuto**: una volta rilevato il gruppo ci si unisce ad esso e si attende la lista dei task;
- **Segnalare l'esecuzione di un task**: attraverso i servizi di comunicazione deve essere possibile avvisare i membri del gruppo dell'esecuzione di un task;
- **Richiedere informazioni su un task**: l'applicazione deve permettere la richiesta di informazioni per eseguire un task;
- **Scambiare messaggi con gli utenti del gruppo**: per meglio coordinarsi è necessario comunicare con gli altri membri del gruppo;
- **Lasciare il gruppo**: una volta terminata l'emergenza l'utente deve poter lasciare il gruppo di primo soccorso.

I casi d'uso sono rappresentati nella figura 5.6.

Funzionamento dell'applicazione

Anche nel caso del prototipo realizzato per ME si può suddividere l'applicazione in interfaccia uomo - macchina e dalla classe che permette di accedere a tutte le funzionalità del framework AGAPE.
Nel caso dell' ME l'interfaccia utente risulta essere più complessa in quanto è costituita da diversi pannelli contenuti nello stesso frame che permettono di eseguire le varie funzioni necessarie alla gestione dell'emergenza.
La necessità di avere un interfaccia di dimensioni limitate per permettere all'applicazione di essere utilizzata su PDA ha portato alla scelta di creare un frame che contiene tutti i pannelli ma che li visualizza solo uno per volta.
In figura 5.7 è rappresentato il diagramma delle classi dell'interfaccia grafica:

Figura 5.6: Casi d'uso per AGAPEmergency ME

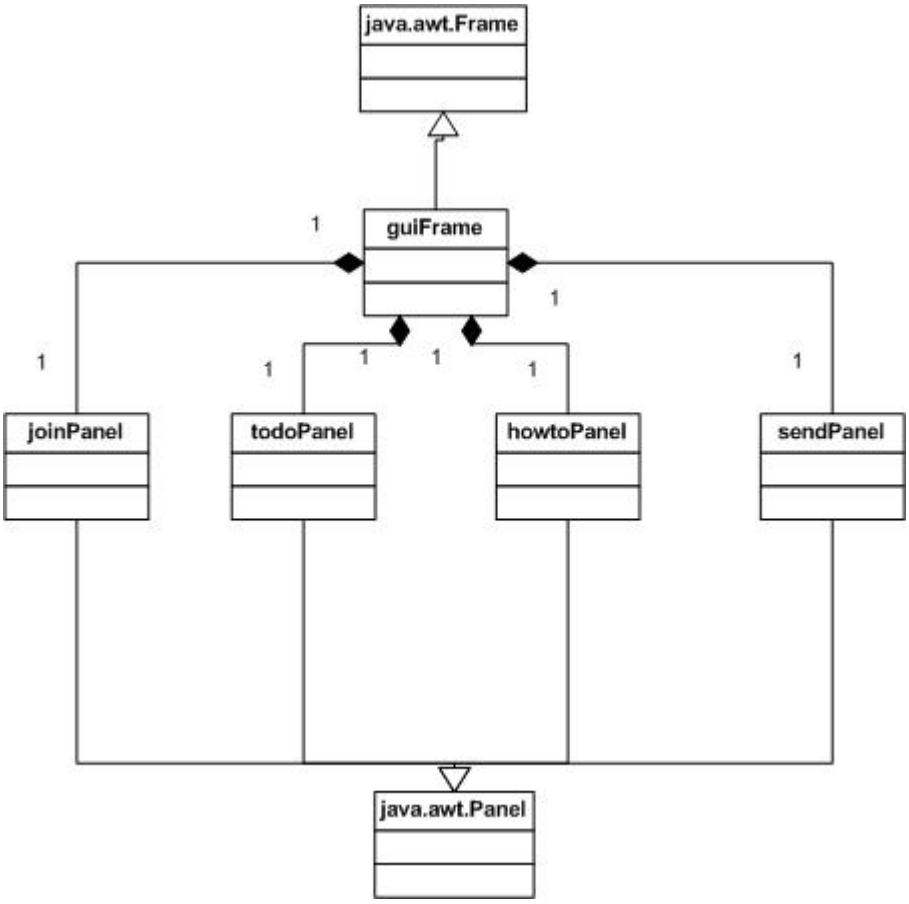

Figura 5.7: Struttura dell'interfaccia grafica

I pannelli vengono visualizzati automaticamente all'occorrenza; ad esempio se arriva un messaggio viene visualizzato automaticamente il todoPanel che contiene l'area di testo in cui vengo riportati i messaggi in ingresso. É comunque possibile cambiare i pannello in modo automatico attraverso il Menu ***View***.

Il sistema viene fatto partire nello stesso modo della versione per LME. Anche in questo caso la classe facade mette a disposizione dei metodi statici per accedere e utlizzare i servizi di AGAPE.

Nelle prossime sezioni sono illustrate le funzionalità di ogni singolo pannello.

5.2.3 Ricezione della richiesta d'aiuto

I dispositivi degli utenti che si trovano nella località sono degli ME. Per questo motivo non sono in grado creare un loro gruppo ma sono in continua ricerca di un gruppo a cui unirsi. Non la richiesta di aiuto viene ricevuta il membro può unirsi al gruppo di primo soccorso agendo sul tasto ***ACCEPT TO HELP*** (5.8) .

Figura 5.8: ME prima e dopo il rilevamento del gruppo di primo soccorso

Dopo essersi unito al gruppo di primo soccorso l'utente riceve una lista di task (*todoList*)da eseguire per soccorrere il malcapitato. Quando viene eseguito un compito è possibile segnalarlo a tutto il gruppo in modo che non venga ripetuto da altri. Una volta selezionato l'utente che ha svolto il task ne diventa il proprietario e solo lui può deselezionarlo. In caso di conflitti LME sceglie come proprietario di un task chi lo ha eseguito e segnalato per primo. Nella figura 5.9 è mostrata la lista e la segnalazione di task eseguito.
Se l'utente ha dei dubbi sull'esecuzione di un determinato task può, mediante il pannello ***HowtoCard*** (rappresentato in figura 5.10) chiedere informazione sui task da eseguire. Se l'informazione è presente sul dispositivo viene subito mostrata a video altrimenti viene inviata una richiesta a tutti gli utenti: chi la riceve e ha l'informazione la invia; se nessuno degli ME ha l'informazione

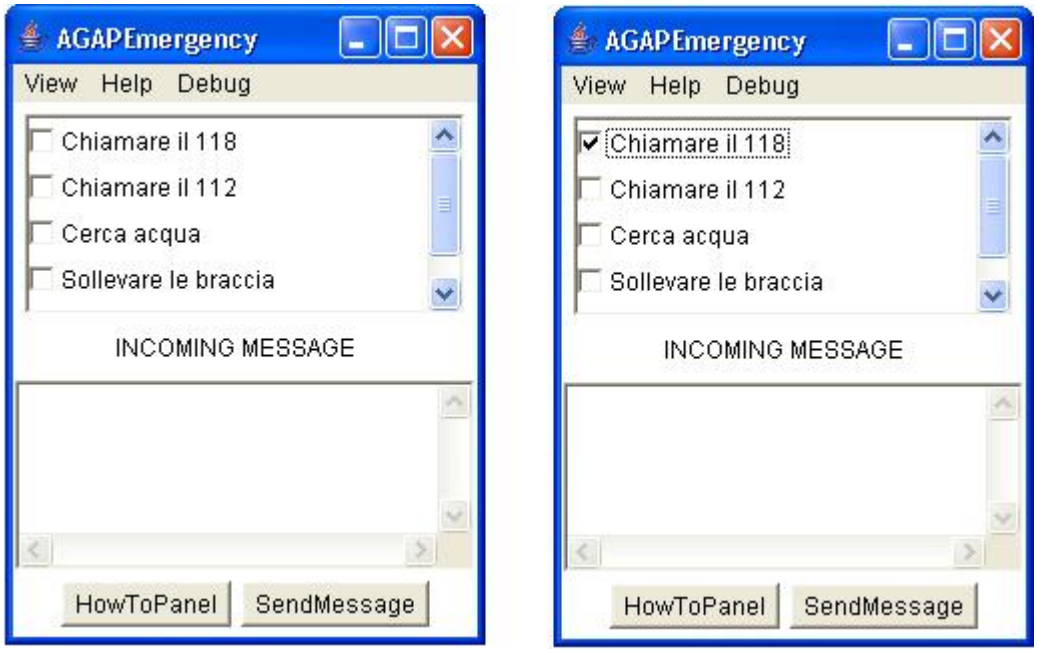

Figura 5.9: Ricezione della lista ed esecuzione di un task

sarà l'LME del malcapitato a inviare la risposta.
La scelta di inviare la richiesta a tutti e giustificata dal fatto che LME potrebbe non essere disponibile e quindi potrebbe non inviare le informazioni richieste. Se qualche ME aveva chiesto la stessa informazione può rispondere e permettere al gruppo di continuare a prestare soccorso anche se la connessione con LME è andata momentaneamente perduta.

La funzionalità più rilevante del prototipo che si è sviluppato è la comunicazione tra gli utenti. Questa funzionalità sfrutta i servizi del Communication Layer sviluppati in questa tesi: **Communcation Service** e **Binding Service**.
L'invio di messaggi avviene utilizzando il pannello ***SendCard*** (figura 5.11) accessibile dal menu **Wiew** o attraverso il pulsante **SendMessage** presente nel ***TodoCard***.
Attraverso questo messaggio è possibile selezionare i destinatari del messaggio. Sono possibili le seguenti opzioni:

ANY PARAMEDICO : permette di mandare un messaggio Any-Cast ad un paramedico;

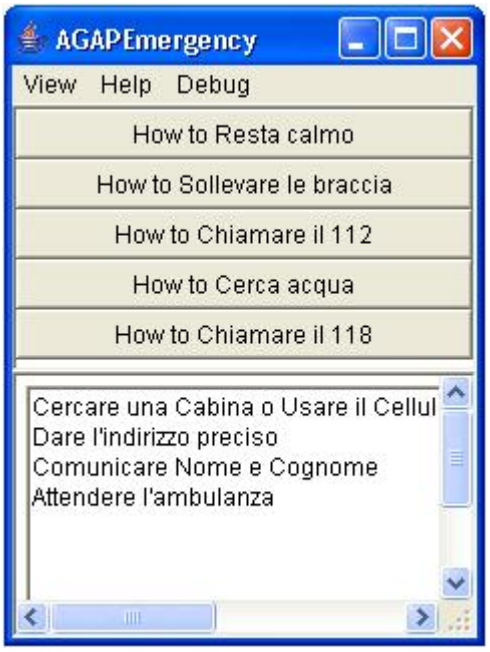

Figura 5.10: Pannello per la richiesta di aiuto nell'esecuzione dei task

Figura 5.11: Pannello per l'invio di messaggi e invio di un messaggio a un paramedico

Figura 5.12: Ricezione di un messaggio

ALL PARAMEDICO : permette di mandare un messaggio in Multi-Cast a tutti i paramedici presenti nella località;

ANY ORDINARY USER : permette di inviare un messaggio Any-Cast ad un persona presente nella località;

ALL ORDINARY USER permette di inviare un messggio in Multi-Cast a tutti gli utenti ordinari della località;

ALL PEOPLE : permette l'invio di messaggi a tutti i membri del gruppo.

La ricezione del messaggio avviene nel pannello in cui è visualizzata la todoList (figura 5.12). In questo modo i messaggi in ingresso possono essere visualizzati anche mentre si sta controllando la lista dei task.

E' da notare che l'applicazione gestisce automaticamente i pannelli. Se arriva un messaggio di testo automaticamente viene caricato la ***todoCard*** e il messaggio viene visualizzato; se viene richiesto un howto e poi si passa ad un altro pannello, quando viene ricevuto il messaggio contenete l'howto la***howtoCard*** viene automaticamente caricata e mostrata a video.

Infine una volta terminata l'emergenza gli ME possono lasciare il gruppo di prima emergenza attraverso il tasto **LEAVE EMERGENCY** e l' LME può chiudere il gruppo attraverso il tasto **EMERGENCY FINISHED** (figura 5.13).

JoinPanel

Attraverso questo pannello è possibile effettuare il join o il leave al gruppo.

Figura 5.13: Ricezione di un messaggio

Appena l'applicazione viene avviata un thread interroga a intervalli di tempo regolari il J/LMS per effettuare il discovering dei gruppi. Appena il gruppo viene rilevato il pulsante verde del pannello viene attivato ed e possibile fare il join al gruppo. Allo stesso modo appena il join va a buon termine il sistema disabilita il tasto verde e abilita il tasto per effettuare il leave del gruppo.
Sia il join che il leave vengono effettuati utilizzando due metodi della classe facade che a loro volta si interfacciano con il J/LMS.

TodoPanel

Una volta uniti al gruppo il listener dell'applicazione si registra presso il CS per ricevere i messaggi in ingresso. Appena LME rileva la presenza del nuovo membro gli invia la todoList. Quando la lista è ricevuta essa viene visualizzata nel TodoPanel ed è pronta per essere utilizzata selezionando i task che si sono eseguiti. Un'altra funzione di questo panello e quella di inviare attraverso un metodo di facade, gli update gli aggiornamenti che l'utente effettua alla lista.
Su questo pannello vengono inoltre riportati i messaggi di testo ricevuti.

HowtoPanel

In questo pannello vengono richieste e visualizzate le informazioni riguardanti lo svolgimento di ogni singolo task.
L'utente può richiedere le informazioni inviando un messaggio a tutti i mem-

bri del gruppo che risponderanno solo se possiedono tale informazione. Una volta ricevuta la risposta questa viene memorizzata e visualizzata sull' howtoPanel che, in caso non sia visualizzato, è automaticamente caricato dal sistema.

SendPanel
Attraverso questo pannello è possibile inviare messaggi ai membri del gruppo presenti nella località. Sono previsti due tipi di membri all'interno della rete: i Paramedici e gli altri. É quindi possibile inviare messaggi di tipo anycast e multicast a questi due tipi di utenti o in alternativa a tutti i membri presenti nel gruppo.

5.3 Test e Performance

Visto il campo di applicazione di AGAPE e delle applicazioni che lo utilizzano è importante conoscere e valutare attentamente le performance di ogni singolo servizio; per fare ciò è stata realizzata un classe di prova che crea dei membri fittizi con profili di diversa lunghezza e che richiede la creazione di Binding sia di tipo AnyCast che di tipo AnyCast e valuta parametri come l'occupazione di memoria del servizio BS a run-time, le dimensioni della Binding Table a fronte di varie richieste e il tempo di risposta. Di seguito verranno presentati attraverso grafici e tabelle i dati ricavati dalla simulazione su un Laptop con processore Intel Pentium IV con 256 MB di Memoria RAM e una frequenza di clock pari a 2,4 GHz.

5.3.1 Tempo di Creazione dei Binding

Come prima prova si è simulata la creazione di un BindingAnyCast utilizzando un numero variabile di membri.
Il profilo dei membri è stato creato appositamente in modo da soddisfare il matching con i vincoli; questo è infatti il caso peggiore: il Binding Service deve infatti inserire tutti i membri nel TMS e nella Tabella.
I profili sono stati creati utilizzando il metodo ***generateView***. Utilizzano

un profilo con due attributi di tipo stringa delle dimensioni di 1258 byte si sono ottenuti i risultati riportati nella seguente tabella:

NUMERO DI MEMBRI	**TIPO DI HANDLER**	**TEMPO DI CREAZIONE [ms]**
10	ANYCAST	320
20	ANYCAST	461
30	ANYCAST	962
40	ANYCAST	921
50	ANYCAST	1122
60	ANYCAST	1342
70	ANYCAST	1562
80	ANYCAST	1793
90	ANYCAST	2013
100	ANYCAST	2243
10	MULTICAST	230
20	MULTICAST	441
30	MULTICAST	661
40	MULTICAST	871
50	MULTICAST	1072
60	MULTICAST	1292
70	MULTICAST	1502
80	MULTICAST	1723
90	MULTICAST	1973
100	MULTICAST	2203

La figura 5.14 rappresenta graficamente i dati ottenuti:

Si è eseguito lo stesso test utilizzando profili rispettivamente con 4 attributi e dimensione 1422 byte e con 8 attributi e dimensione 1748 byte; nella seguente tabella sono riportati i dati ottenuti:

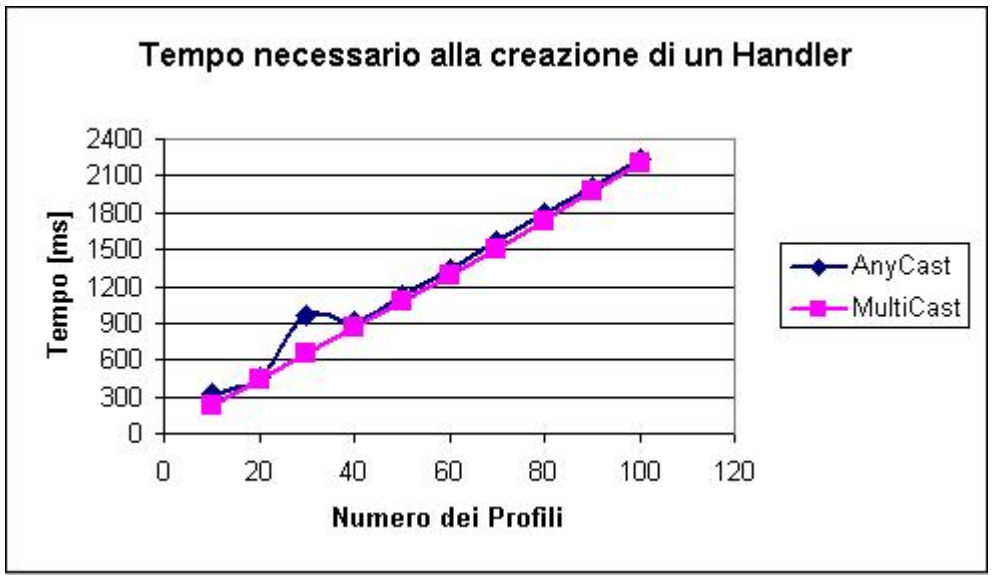

Figura 5.14: Tempo necessario alla creazione di un Handler con profili di 2 Attributi

NUM. MEMBRI	**TIPO DI HANDLER**	**TEMPO [ms] 2 Attr**	**TEMPO [ms] 4 Attr**
10	ANYCAST	641	961
20	ANYCAST	802	1523
30	ANYCAST	1261	2313
40	ANYCAST	1672	3065
50	ANYCAST	2073	3825
60	ANYCAST	2474	4587
70	ANYCAST	2894	5338
80	ANYCAST	3305	6098
90	ANYCAST	3715	6849
100	ANYCAST	4126	7641
10	MULTICAST	361	751
20	MULTICAST	871	1482
30	MULTICAST	1182	2223
40	MULTICAST	1602	3015
50	MULTICAST	2013	3755
60	MULTICAST	2414	4477
70	MULTICAST	2804	5238
80	MULTICAST	3215	5988
90	MULTICAST	3615	6769
100	MULTICAST	4006	7540

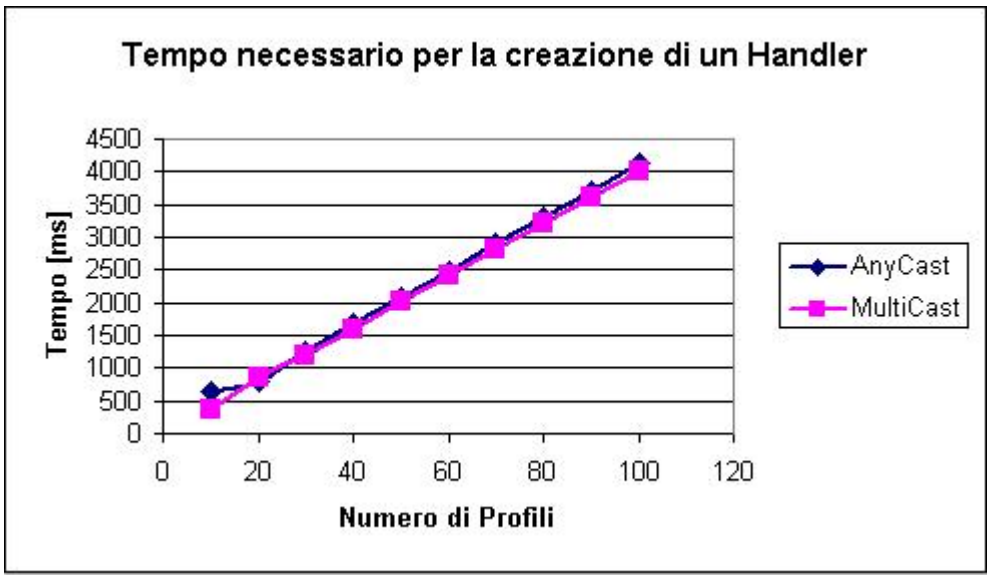

Figura 5.15: Tempo necessario alla creazione di un Handler con profili di 4 Attributi

I due grafici sono riportati rispettivamente nelle figure 5.15 e 5.16:
Un ulteriore test per quanto riguarda i tempi di creazione di un Handler è quello di creare un Binding con strategia EarlyBinding quando il TMS è già presente nella BindingTable a seguito di una precedente richiesta.
A titolo esemplificativo si riportano i dati relativi ad un test con profili con 8 attributi di dimensione pari a 1748 byte per la creazione di un MulticastHandler che trova nella BindingTable un TMS già valido:

Numero di Profili	**Senza TMS**	**con TMS**
10	1021	50
20	1592	70
30	2013	70
40	2033	180
50	2364	170
60	3155	170
70	3936	170
80	4717	180
90	5478	180
100	5628	180

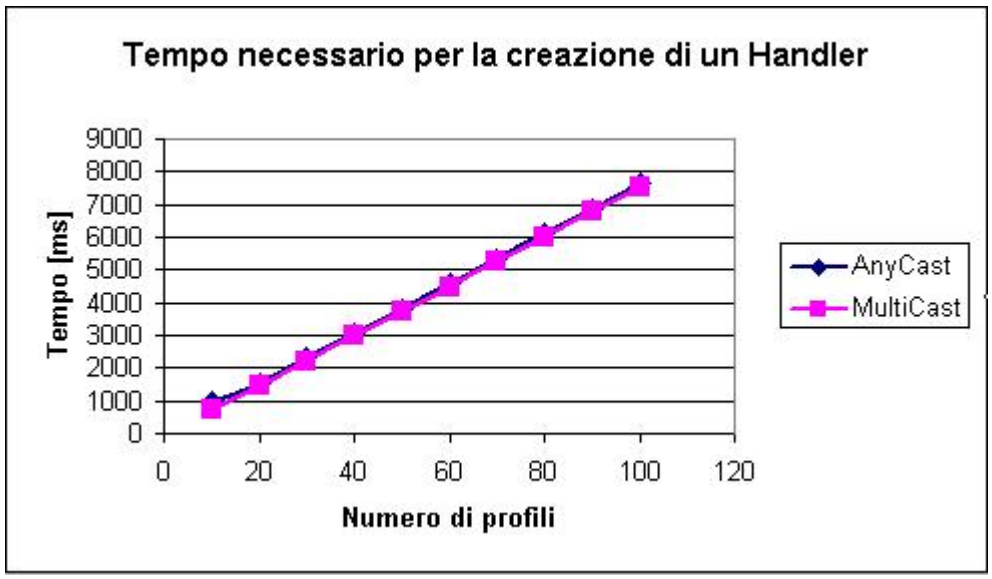

Figura 5.16: Tempo necessario alla creazione di un Handler con profili di 8 Attributi

Si nota subito che quando il TMS contenuto nella Binding Table è valido il tempo richiesto al Bindind Service per creare l'handler è notevolmente inferiore; quindi l'uilizzo della Binding Table per memorizzare i binding già effettuati risulta una buona scelta.

5.3.2 Dimensioni di un Binding

Le dimensioni di un binding variano a seconda dei seguenti fattori:

- ***Dimensioni della Vista***: quanti membri ci sono all'interno della località;
- ***Dimensioni dei Profili***: quanti attributi possiede ogni membro della vista;
- ***Criteri di selezione***;

Per quanto riguarda i primi due punti è banale verificare che al crescere dei membri di una località sarà necessario filtrare più profili e, necessariamente più attributi hanno i profili, maggiore sarà la dimensione che occuperanno nel TMS.
Il terzo punto dipende dalla tipologia dell'applicazione: se infatti ci si trova

in un contesto in cui operano differenti tipologie di utenti e se ne vuole selezionare un particolare tipo il TMS sarà relativamente piccolo se confrontato con l'intera vista.
Ad esempio se la vista è composta da 100 membri e nel nostro contesto operativo sono presenti solo *Ingegneri* e *Matematici* equamente distribuiti è naturale che nel caso in cui un membro voglia ottenere un binding con un *Matematico* otterrà un TMS di 50 membri; se invece nel nostro contesto applicativo sono presenti anche *Fisici*, *Chimici* e *Astronomi* il TMS risulterà al più di 20 membri.

Conclusioni

Lo sviluppo di servizi collaborativi avanzati in scenari MANET solleva problemi complessi che richiedono l'adozione di nuove linee guida per il design e l'implementazione dei servizi. In questa tesi abbiamo mostrato come la possibilità di beneficiare della visibilità delle caratteristiche e degli attributi dei membri dei gruppi possa facilitare la collaborazione in scenari dinamici di Mobile Ad-Hoc Network.

In questa tesi è stato mostrato un supporto innovativo per la comunicazione di gruppo context-aware che abbiamo progettato ed implementato. In particolare, questo supporto consente di individuare i partner per la collaborazione in base alle loro caratteristiche e di creare e riqualificare dinamicamente i binding fra di essi in base alla corrente situazione di rete. Infine, per valutare l'applicabilità del supporto alla comunicazione di gruppo realizzato, è stato mostrato il prototipo di un'applicazione per il primo soccorso in scenari mobile ad-hoc network.

Il prototipo realizzato ha confermato che il supporto alla comunicazione di gruppo che abbiamo implementato è adatto allo sviluppo di servizi collaborativi in contesti MANET. I primi incoraggianti risultati ottenuti possono stimolare ulteriori attività di ricerca al fine di supportare lo scheduling dei messaggi e l'adattamento del loro formato in base alle caratteristiche dei terminali utente.

Appendice A

Installazione e configurazione del prototipo su PDA

Lo scopo di questa appendice è di descrivere la configurazione e l'installazione di AGAPEmergency su PDA.
I dispositivi utilizzati sono due PDA HP IPAQ: uno della serie 3700 e il secondo della serie 5500. Per quanto riguarda il primo modello è stato necessario utilizzare una scheda wireless esterna mentre il secondo è dotato di connettività wireless integrata.
Il sistema operativo utilizzato è stato Windows CE mentre la macchina virtuale utilizzata è stata la **Jeode***Runtime* prodotta da insigna. Questa macchina virtuale è compatibile perfettamente con le specifiche di PersonalJava 1.2 delle Sun.
Una volta installata la macchina virtuale attraverso il software ActiveSync è possibile avviare la classe contenete il main o l'archivio jar in due modi differenti:

Attraverso la riga di comando di evm : selezionare l'icona EVM (Start, Programmi, Jeode) e immettere le opzioni nel prompt
evm: [opzioni] nomeClasse [argomenti]

Attraverso un link a evm : creare un collegamento (.lnk) alla macchina virtuale e trasferirlo modificarlo tramite un editor inserendo i parametri necessari all'esecuzione di un classe.

La distribuzione di AGAPEmergency realizzata per IPAQ è così composta:

agapelib.jar : rappresenta il framework AGAPE e contiene tutti i suoi servizi;

com : è la directory contenente classi utili alla gestione dei profili;

javax : è un' altra directory contenente altre classi utili alla gestione dei profili;

javax : contiene i file di configurazione di AGAPE;

prototipo : è la directory che rappresenta il package del prototipo realizzato.

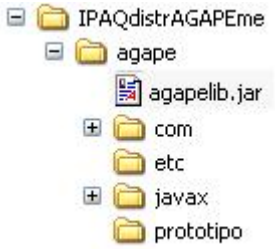

Figura A.1: Struttura della distribuzione di AGAPEmergency per PDA

Per lanciare l'applicazione si è creato un link alla macchina virtuale e si è modificato attraverso un editor di testi; il contenuto del file dopo l'aggiunta dei parametri e delle opzioni è il seguente:

18#\Windows\evm.exe -Djeode.evm.console.local.keep=TRUE -classpath \agape\agapelib.jar;\agape\ prototipo.prototipoMain

Attraverso -Djeode.evm.console.local.keep=TRUE la macchina virtuale lascia aperta la console anche quando il programma ha terminato in modo da poter vedere eventuali messaggi di errore.
-classpath indica dove si trovano le classi mentre prototipo.prototipoMain indica quale classe eseguire.
Il file cosi creato viene inserito nella cartella agape e il tutto viene trasferito nella root del PDA: \agape.
Se la posizione viene modificata è importante lanciare l'applicazione settando la proprietà agape.base al percorso in cui si trova l'archivio agapelib.jar; questa operazione può essere eseguita attraverso l'opzione -Dagape.base={percorso}.

Figura A.2: AGAPEmergency in esecuzione su PDA

Nelle seguenti foto si vede l'applicazione AGAPEmergency in esecuzione su un PDA IPAQ:

Rigraziamenti

Non è possibile ricordare tutti quelli che mi hanno dato una mano per arrivare a questo giorno.
Un ringraziamento va a tutti i professori che in questi anni di università mi hanno permesso di accrescere le mie conoscenze, in modo particolare il Prof. Antonio Corradi che mi ha dato la possibilità di realizzare questa tesi.
Sono infinitamente grato all' Ing. Dario Bottazzi per la sua pazienza, per la sua disponibilità e per i consigli che mi ha dato durante tutto lo svolgimento della tesi.
Ringrazio Marco, Andrea e Michele per avermi aiutato e consigliato durante la realizzazione del software e della tesi. Grazie a Ilias per aver messo a disposizione i sorgenti LATEX della sua tesi.
Molte grazie a tutti i responsabili del lab2, in particolare all' Ing. Fabio Bucciarelli per aver sopportato le mie continue richieste.
Un grazie va anche ai miei genitori che mi hanno sostenuto in questi anni di studi anche se non sempre mi sono dimostrato riconoscente.
Un ringraziamento particolare va ad Elisabetta, la mia ragazza da sempre e per sempre che più di ogni altro mi è stata vicina durante questi anni di università ed è comunque riuscita a sopportarmi.
Naturalmente ringrazio tutti coloro che hanno contribuito a correggere questo lavoro.

Bibliografia

[1] A.Mukherjee, S.Bandyopadhyay, D.Saha, *Location Management and Routing in Mobile Wireless Networks*

[2] M.Ilyas, *The Handbook of Ad Hoc Wireless Networks*

[3] A.S.Tanenbaum, *Reti di computer*

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

[5] D.Bottazzi, A.Corradi, R.Montanari, *Context-Aware Group Communication in Mobile Ad-Hoc Networks*
http://www.lia.deis.unibo.it/research/AGAPE/Docs/RYFGNY5K5YLQ6Q90.pdf

[6] D.Bottazzi, A.Corradi, R.Montanari, *Enabling Context-Aware Group Collaboration in MANETs*

[7] D.Bottazzi, A.Corradi, R.Montanari, *Context-Awareness for Impromptu Collaboration in MANETs*
http://www.lia.deis.unibo.it/research/AGAPE/Docs/01383399.pdf

[8] D.Bottazzi, A.Corradi, R.Montanari, *AGAPE: A Location-Aware Group Membership Middleware for Pervasive Computing Environments*
http://www.lia.deis.unibo.it/research/AGAPE/Docs/pap-deis-18.pdf

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

[10] C.Caini, *Bluetooth* disponibile su
http://www3.deis.unibo.it/Staff/Research/CCaini/bluetooth.htm

[11] Sito web della distribuzione Linux per PDA: http://familiar.handhelds.org/

[12] Sito web contenete informazioni su java per PDA: http://www.comp.lancs.ac.uk/computing/users/fittond/ppcjava.html