Progetto PERMESSO

PERsistent MESSagging in ad hOc networks

Manuela Bassetti – matr. 0000260504
C.d.L.S. Ingegneria Informatica
Università di Bologna
***Reti di Calcolatori LS** – Prof. Antonio Corradi – AA. 2006-07*
Tutor del progetto: Eugenio Magistretti

Abstract

Permesso è un servizio di messaggistica persistente in MANET (Mobile Ad Hoc Networks), pensato per dispositivi portatili, quali laptop, pda o smartphone.

1 Introduzione

1.1 Le MANET

Le reti mobili ad hoc (MANET, Mobile Ad Hoc Networks) sono reti wireless costituite da dispositivi portatili che possono liberamente e dinamicamente stabilire una comunicazione in assenza di una infrastruttura fissa, organizzandosi in topologie di rete "ad-hoc".

L'utilizzo di queste reti porta notevoli vantaggi, tra i quali la proprietà di non richiedere un server centrale né la presenza di access point. In assenza di un controllo centrale saranno quindi i nodi stessi a organizzarsi in modo autonomo, e a gestire i cambiamenti delle topologia della rete, dovuti alla dinamicità della rete stessa. A causa dell'instabilità della comunicazione wireless e della possibilità di movimento degli utenti sono frequenti perdite di connessione e ingressi di nuovi nodi.

È inoltre necessario valutare semantica e proprietà della comunicazione, in quanto i sistemi sono inevitabilmente orientati a dinamiche di tipo best-effort, con frequenti perdite di messaggi e cadute dei nodi.

1.2 I dispositivi

Nello sviluppo del progetto è stato necessario considerare alcune caratteristiche dei dispositivi utilizzati. Si tratta infatti di dispositivi con memoria limitata (espandibile esternamente, a fronte di un aumento dei tempi di accesso), ridotte risorse run-time, e alimentazione a batteria che porta ad avere autonomia limitata.

1.3 Lo stato dell'arte e gli strumenti utilizzati

Le specifiche di progetto richiedevano l'utilizzo della J2ME, una java virtual machine ridotta utilizzabile su palmari, di cui esistono due configurazioni: CLCD, più leggera, e CDC, adatta a dispositivi di fascia più alta. Per lo svolgimento del progetto è stata scelta la configurazione CDC 1.1, e il Personal Profile.

Sono attualmente disponibili diverse versioni della J2ME, alcuni emulatori che permettono le simulazioni sui laptop e alcuni tool di sviluppo, ma si tratta per la maggior parte dei casi di soluzioni proprietarie, incompatibili tra loro. In seguito all'analisi del software disponibile abbiamo scelto la j9 di IBM, l'emulatore Nokia Series 80 e il NetBeans CDC Mobility Pack 5.5.1 su piattaforma NetBeans 5.5.1. Queste scelte sono state rese necessarie, per

avere la possibilità di sviluppare un'applicazione compatibile con le diverse tipologie di dispositivi e sistemi operativi.
Si è rivelata necessaria anche la ricerca di un database adatto a dispositivi portatili, per il quale non fosse necessaria una lunga installazione e configurazione. A tale scopo è stato scelto il db4o.

2 Permesso

2.1 Servizi principali

Permesso desidera sfruttare la dinamicità intrinseca nel modello di rete MANET per fornire agli utenti soluzioni di instant messaging. Nel progetto vengono sviluppati tre servizi principali.

- **Discovery**
  Servizio che si occupa di gestire l'entrata e l'uscita dei nodi nella rete, ponendo particolare attenzione quindi sullo stato dei nodi.

- **Chatting sincrono**
  Servizio che gestisce lo scambio di messaggi fra due utenti presenti contemporaneamente nella MANET. Possono essere attivate anche più chat sincrone con diversi utenti contemporaneamente.

- **Chatting asincrono**
  Servizio che permette ad un utente presente sulla MANET di lasciare messaggi per un utente attualmente non presente. La consegna di tali messaggi è delegata al supporto e può essere realizzata in due modalità. La prima prevede la presenza nella rete di un nodo centrale con memoria illimitata, denominato Persistent Server, che memorizza i messaggi e li consegna al destinatario al suo ingresso nella MANET. Utilizzando la seconda modalità i messaggi saranno consegnati agli altri nodi presenti sulla rete. Poiché i nodi hanno capacità limitata tale servizio sarà best-effort, quindi si procederà alla cancellazione dei messaggi più vecchi se necessario. In caso di caduta del nodo i messaggi da lui memorizzati andranno persi.

2.2 Estensioni realizzate

- **Presenza**
  Per gestire l'eventuale caduta di un nodo è stato introdotto un servizio di presenza con cui il nodo notifica, a intervalli fissati, la sua presenza sulla MANET.

- **Amicizia**
  Servizio che permette di stabilire una relazione di amicizia con un altro nodo della MANET. Per questo servizio si prevedono due modalità: amicizia per UUID, che permette di stabilire una relazione di amicizia con un nodo specifico, e amicizia per profilo in cui viene effettuata una ricerca di amici sulla base di un profilo specificato.
  È inoltre fornita la possibilità di rimuovere un amico, sia che questo sia presente attualmente sulla rete, sia che non lo sia.

- **Persistent Server "dinamico"**
  Per il servizio di chatting asincrono veniva richiesta la presenza o meno del Persistent Server. Con questa estensione ne vengono gestiti ingresso e uscita in modo dinamico.

La comunicazione tra i dispositivi avviene tramite l'interfaccia wireless 802.11.
Le specifiche progettuali richiedono lo sviluppo di protocolli ah hoc che non sfruttino middleware preesistenti, questo rende possibile la realizzazione di un'applicazione molto leggera, adatta

a dispositivi portatili, anche se complica lo sviluppo.
Per la realizzazione del progetto sono state fatte alcune assunzioni: assumiamo che tutti i dispositivi presenti sulla MANET siano a distanza di comunicazione 1-hop e in reciproca visibilità; la seconda assunzione è che, una volta entrati nella rete, gli indirizzi dei dispositivi rimangono fissi. Consideriamo inoltre la rete MANET preesistente.

2.3 Servizi presentati

Nei prossimi paragrafi saranno approfonditi il servizio di Chatting asincrono in assenza di Persistent Server e la ricerca di amici tramite profilo.
Prima di entrare nel merito dei servizi specifici è bene chiarire che abbiamo scelto di utilizzare una porta fissa e stabilita a priori per ogni servizio, in modo da realizzare un'applicazione più leggera possibile. Naturalmente in caso di dense MANET questa scelta potrebbe portare a forti sequenzializzazioni, ritardi e congestioni. Per risolvere tali problemi è tuttavia sufficiente destinare più porte per lo stesso servizio in modo da bilanciare il carico.

3 Chatting asincrono

3.1 Struttura generale

Il servizio di chatting asincrono realizzato si basa sullo scambio di messaggi contraddistinti da un tipo, ognuno dei quali ha un preciso scopo e significato.
Le tipologie di messaggi utilizzati sono le seguenti:

- OFFLINE_MESSAGES, che rappresenta un messaggio inviato a un amico attualmente offline
- ACK_OFFLINE_MESSAGES, ack che comunica la ricezione del messaggio per un amico offline
- REQUEST_OFFLINE_MESSAGE, con cui un nodo chiede, successivamente al discovery iniziale, di ricevere i messaggi che gli sono stati inviati quando era offline
- SEND_OFFLINE_MESSAGE, con cui un nodo in uscita chiede di inviare al Persistent Server i messaggi per i contatti offline
- ACK_PERSISTENT_MESSAGE, ack inviato dal Persistent Server per segnalare la propria presenza sulla rete e quindi la disponibilità a ricevere i messaggi offline dal nodo in uscita
- ACK_NODE_MESSAGE, ack inviato dai nodi per segnalare la propria disponibilità a ricevere i messaggi offline dal nodo in uscita
- REQUEST_MEMORY_MESSAGE con cui un nodo in uscita chiede agli altri nodi se hanno possibilità di memorizzare e quindi ricevere messaggi offline
- BLOCK_MEMORY_MESSAGE, con cui un nodo in uscita chiede ai nodi a cui intende inviare i propri messaggi offline di riservare parte della loro memoria per memorizzare tali messaggi
- ACK_BLOCK_MEMORY_MESSAGE, con cui un nodo accetta di riservare parte della propria memoria disponibile per memorizzare i messaggi offline del nodo in uscita

Il servizio si compone di due fasi: al discovery il nodo entrante chiederà di ricevere i messaggi offline che sono stati consegnati per lui, mentre all'uscita il nodo uscente chiederà di inviare i messaggi per i destinatari offline.
La consegna all'ingresso dei nodi e la ricezione all'uscita potranno essere gestite dal Persistent Server se presente, oppure dagli altri nodi della rete; in questa relazione tratteremo la parte

relativa alla consegna e ricezione in assenza di Persistent Server.

3.2 Recapito dei messaggi all'ingresso

Quando un nodo compare sulla rete effettuando il Discovery (v. servizio di Discovery) ogni altro nodo deve verificare se ha memorizzato dei messaggi asincroni per lui, sia propri che da parte di altri nodi. La risposta del Discovery conterrà il numero di messaggi offline per tale nodo.

Il nodo in entrata invierà ai nodi che hanno messaggi asincroni per lui (che potranno essere nodi amici o sconosciuti) la richiesta per ricevere tali messaggi, tramite un messaggio REQUEST_OFFLINE_MESSAGE.

Il nodo che riceve la richiesta inizia a inviare in ordine i messaggi tramite OFFLINE_MESSAGES, attendendo ack singoli (ACK_OFFLINE_MESSAGES) prima del successivo invio.

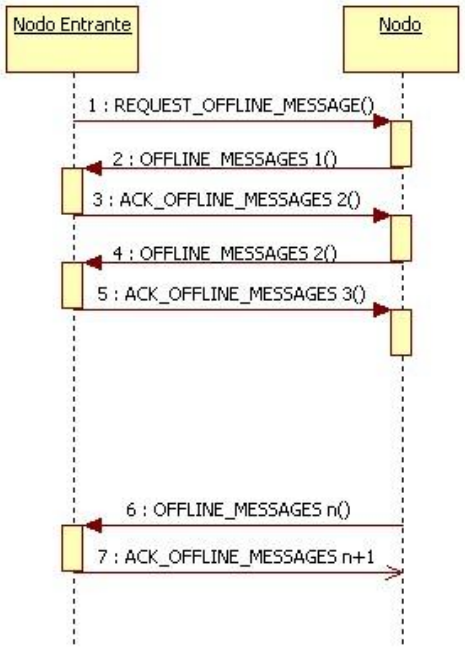

Figura 1 Recapito dei messaggi asincroni

Nel caso in cui, in seguito alla richiesta iniziale di invio dei messaggi offline, non iniziasse la ricezione dei messaggi stessi entro un certo timeout, la richiesta verrebbe ritrasmessa fino a due volte.

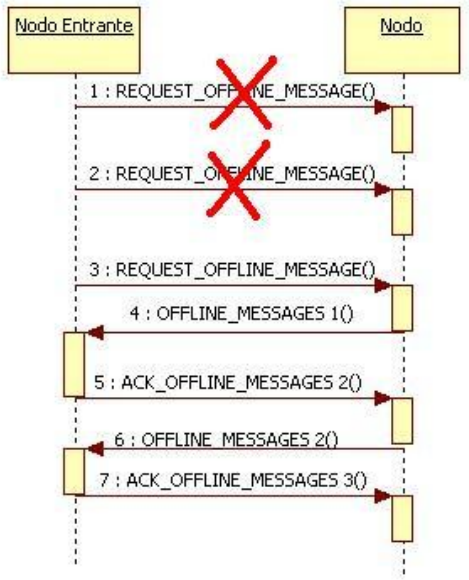

Figura 2 Perdita della richiesta dei messaggi

In caso di ulteriore perdita della richiesta il protocollo viene terminato e i messaggi potranno essere richiesti successivamente, alla ricezione dei messaggi di presenza nella rete.

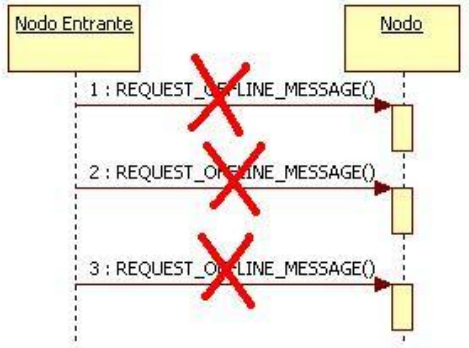

Figura 3 Perdita tripla della richiesta

Nel caso un ack atteso non sia ricevuto viene effettuato il rinvio del messaggio fino a tre volte o fino allo scadere di un timeout di ricezione. Al termine del timeout i messaggi verranno cancellati.

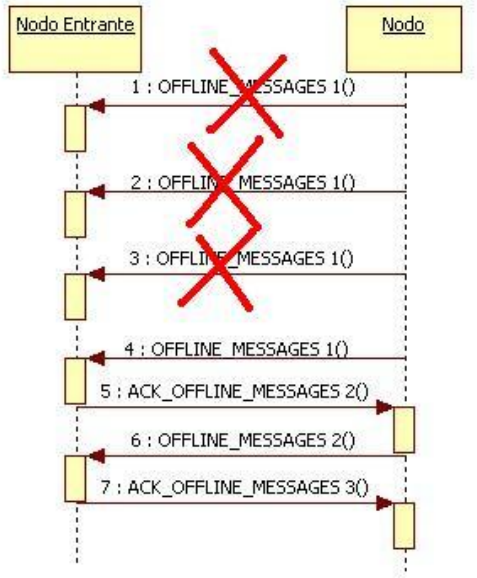

Figura 4 Perdita di tre messaggio offline

I messaggi asincroni ricevuti possono essere testuali o di cancellazione. I messaggi testuali saranno visualizzati in un pannello di conversazione da cui non sarà possibile inviare alcun messaggio.

Figura 5 Ricezione messaggi offline

Un messaggio di cancellazione dice invece che il contatto che ce lo ha inviato ci ha cancellati, di conseguenza verrà avviata la procedura per cancellare il mittente del messaggio, in modo da mantenere coerenza tra le liste di amici.

3.3 Consegna dei messaggi in uscita

È possibile scrivere messaggi anche a destinatari attualmente non presenti nella rete. Per farlo si utilizza la stessa form utilizzata per le conversazioni sincrone. I messaggi non vengono inviati immediatamente, ma memorizzati, e consegnati al destinatario, quando questi compare sulla rete.
Tuttavia un nodo potrebbe decidere di uscire dalla rete prima che il destinatario dei messaggi offline sia comparso. In questo caso i messaggi saranno recapitati al Persistent Server, che si occuperà dell'inoltro, o, in sua assenza, agli altri nodi della rete.

Esamineremo qui il caso in cui il Persistent Server non sia presente nella rete.
In fase di uscita il nodo dovrà inviare in broadcast un messaggio REQUEST_MEMORY_MESSAGE, con il quale chiede a ogni nodo della rete se ha memoria sufficiente per ospitare i messaggi. Questa richiesta è dovuta al fatto che si è deciso di limitare il numero di messaggi memorizzabili su ogni nodo, poiché non si dispone di dispositivi a capacità illimitata.
In seguito a questa richiesta è possibile ricevere un messaggio ACK_PERSISTENT_MESSAGE, in caso il Persistent Server sia presente nella rete; in questo caso viene avviata la procedura di ricezione dei messaggi dal Persistent Server.
In caso invece vengano ricevuti solo ACK_NODE_MESSAGE, ai nodi che hanno risposto dando disponibilità di memoria verrà inviato un messaggio BLOCK_MEMORY_MESSAGE, specificando il numero di messaggi che dovranno essere memorizzati. Il messaggio per richiedere il blocco della memoria verrà inviato ai due nodi che dichiarano di avere maggiore capacità disponibile, in modo da mandare i propri messaggi a un nodo e i messaggi di altri ad un altro nodo. Tuttavia se uno dei due nodi ha spazio sufficiente per ricevere sia i messaggi propri che quelli altrui verranno inviati tutti a lui. Si ricorrerà ai nodi che dichiarano minore capacità solo nel caso in cui nessun nodo avesse capacità sufficiente o se i

nodi che dichiarano capacità maggiore non dovessero rispondere con un ack al messaggio di blocco della memoria.
Il numero di messaggi da memorizzare potrebbe essere quindi maggiore del numero di messaggi effettivamente memorizzabili dal nodo che ha inviato la propria disponibilità; in tal caso, sul nodo ricevente, dovranno essere cancellati i messaggi più vecchi per permettere la memorizzazione dei messaggi che il nodo uscente deve inviare.
In seguito alla ricezione di un messaggio di conferma, ACK_BLOCK_MEMORY_MESSAGE potrà iniziare l'invio vero e proprio dei messaggi, per ognuno dei quali si attenderà la ricezione di un ack singolo.

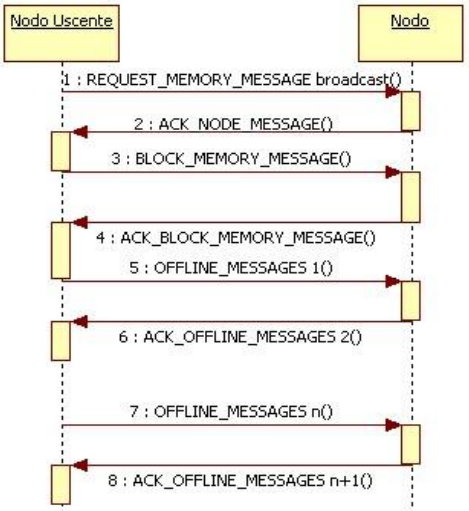

Figura 6 Protocollo di invio messaggi offline in uscita

Se non si riceve alcuna risposta in seguito all'invio del messaggio REQUEST_MEMORY_MESSAGE entro un certo timeout si uscirà dall'applicazione senza inviare i messaggi offline, in quanto è prevedibile che la rete sia estremamente congestionata o non ci siano altri nodi presenti.
Se non si riceve risposta al BLOCK_MEMORY_MESSAGE ci si rivolgerà agli altri nodi che hanno dato la propria disponibilità, ma se nessuno dei nodi dovesse rispondere si uscirà senza inviare i messaggi.

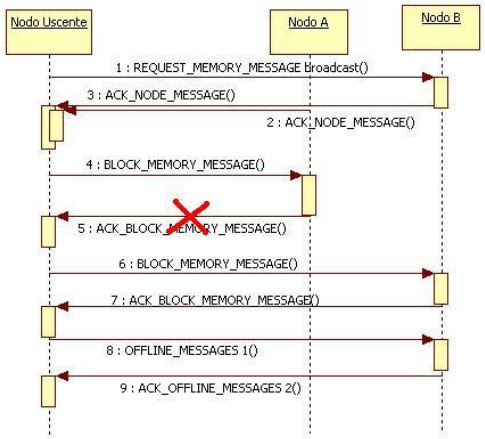

Figura 7 Perdita dell'ACK_BLOCK_MEMORY_MESSAGE

La mancata ricezione di un ack che confermi l'arrivo di un messaggio offline porterà invece alla ritrasmissione del messaggio, fino a tre volte. Se non si riceverà l'ack neanche alla terza ritrasmissione si uscirà dall'applicazione senza inviare i messaggi offline non ancora mandati.

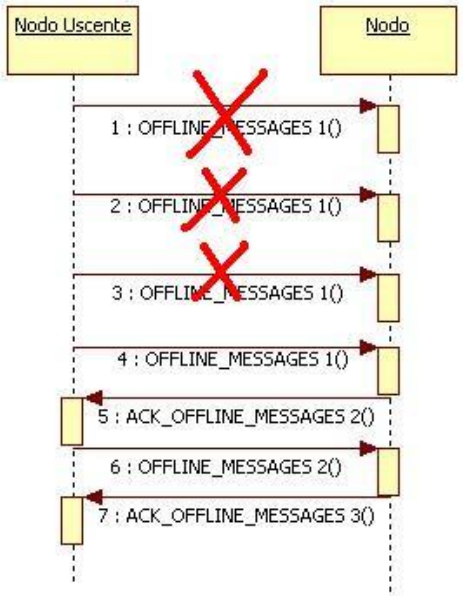

Figura 8 Perdita di tre messaggio offline

Il ricevente, una volta terminata la ricezione dei messaggi offline, li memorizzerà in un database locale, in attesa di consegnarli ai rispettivi destinatari.

3.4 Aspetti implementativi

La richiesta di ricezione dei messaggi offline, all'ingresso nella rete, è effettuata dal destinatario dei messaggi, tramite un thread attivato al momento del Discovery se i nodi che rispondono dichiarano di avere messaggi per il nodo entrante.
Dal lato dei nodi presenti nella rete invece, sarà presente un thread, attivato all'avvio dell'applicazione, che si occuperà di ricevere le richieste di ricezione dei messaggi offline da parte dei nodi in ingresso.

Per la gestione dell'uscita avremo un thread sul nodo uscente, attivato al momento dell'uscita dall'applicazione, che si occupa di inviare i messaggi di REQUEST_MEMORY_MESSAGE e BLOCK_MEMORY_MESSAGE.
Gli altri nodi presenti nella rete avranno attivato un thread all'avvio dell'applicazione che si occupa di ricevere tali messaggi e di inviare gli ack.

3.5 Test e risultati sperimentali

Sono stati effettuati alcuni test per verificare il corretto funzionamento del servizio implementato. Tali test tuttavia sono stati eseguiti avendo a disposizione solo sei dispositivi, di conseguenza prima di portare il servizio su più larga scala sarebbe opportuno testarlo con un numero maggiore di nodi connessi.
Le probabilità di perdita dei messaggi di richiesta e dei messaggi offline sono piuttosto basse, e diventano quasi nulle grazie all'introduzione delle ritrasmissioni.

Sono stati valutati:

- per la ricezione dei messaggi offline al momento del discovery, il tempo di attesa tra l'invio della richiesta e la ricezione del primo messaggio;
- per la consegna dei messaggi offline al momento dell'uscita, il tempo di attesa tra l'invio del messaggio REQUEST_MEMORY_MESSAGE e la ricezione dell'ack che conferma l'avvenuta consegna del primo messaggio offline.

I tempi di consegna dei messaggi offline, in uscita e in ingresso, calcolati in ms, sono i seguenti:

Consegna dei messaggi offline in uscita	Ricezione dei messaggi offline in entrata
2483	360
2614	165
2484	163

La consegna dei messaggi offline all'uscita richiede maggiore tempo, in quanto sono necessari un numero maggiore di messaggi per la contrattazione iniziale e l'avvio del protocollo, oltre all'elaborazione per il calcolo dello spazio disponibile. In questi tempi non è ovviamente incluso il tempo di salvataggio dei messaggi, che viene effettuato al termine della ricezione.
La ricezione dei messaggi all'ingresso nel sistema richiede invece tempi minori, in quanto non è prevista alcuna elaborazione prima dell'invio vero e proprio dei messaggi. Il tempo di visualizzazione dei messaggi offline ricevuti è trascurabile.

4 Amicizia

4.1 Struttura del servizio

Per poter comunicare con un nodo è necessario avere instaurato una relazione biunivoca di amicizia con lui.

Per questo motivo abbiamo scelto di introdurre un servizio di ricerca di amici, per il quale si prevedono due modalità: ricerca di un amico specifico tramite identificativo, e ricerca di amici sulla base di un profilo specificato.
Questo servizio si basa sullo scambio di messaggi differenziati in base al tipo. I tipi di messaggi utilizzati sono i seguenti:

- FRIENDSHIP_UUID, con cui si ricerca un amico specificando l'identificativo dell'amico cercato
- FRIENDSHIP_PROFILE, che permette di ricercare un amico in base a determinate caratteristiche specificate
- FRIENDSHIP_DELETE, che permette di cancellare un amico, sia che sia attualmente presente sulla rete, sia che non lo sia
- FRIENDSHIP_ACK, che permette di rispondere a una richiesta di amicizia, accettandola o rifiutandola

4.2 Amicizia per profilo

In caso di ricerca per profilo l'utente potrà ricercare uno o più amici, specificando tramite un'apposita form il profilo richiesto dai potenziali amici, e il numero massimo di amici che intende aggiungere.

Figura 9 Form di inserimento del profilo cercato

La richiesta contenente il profilo ricercato, cioè un messaggio di tipo FRIENDSHIP_PROFILE, sarà inviata in broadcast a ogni nodo della MANET. In seguito a una verifica del match tra il profilo richiesto e il profilo del ricevente, se compatibili, verrà chiesto all'utente se desidera o meno stabilire l'amicizia.
In caso affermativo verrà inviata la disponibilità dell'utente con un messaggio FRIENDSHIP_ACK, e si attenderà una ulteriore conferma, sempre tramite un messaggio FRIENDSHIP_ACK, dal nodo che ha inviato la richiesta iniziale. Questa conferma finale è stata introdotta in quanto non è possibile sapere quanti nodi risulteranno compatibili con il profilo richiesto e invieranno la propria disponibilità. Potrebbero perciò giungere risposte positive da un numero di utenti maggiore di quello richiesto, ma saranno aggiunti nella lista di amici solo i primi utenti che hanno dato la loro disponibilità. A tali utenti verrà ovviamente inviato un ack di conferma di avvenuta aggiunta. Il nodo che ha dato la propria disponibilità dovrà quindi attendere questo ack finale, prima di aggiornare a sua volta la lista dei proprio amici.

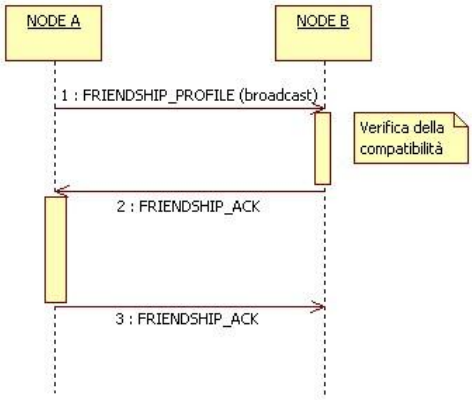

Figura 10 Protocollo di ricerca di amici per profilo

I messaggi di disponibilità per lo stabilire dell'amicizia oltre il numero massimo richiesto saranno ignorati e il protocollo terminerà senza l'aggiunta dell'amico da entrambe le parti:

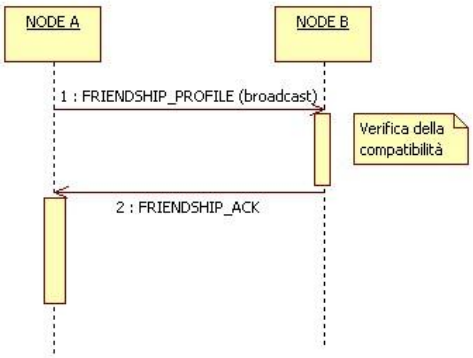

Figura 11 Troppe risposte alla richiesta di amicizia: le ignoriamo

Il nodo che riceve la richiesta la ignorerà non inviando il messaggio FRIENDSHIP_ACK nel caso in cui il profilo ricevuto non sia compatibile con il proprio:

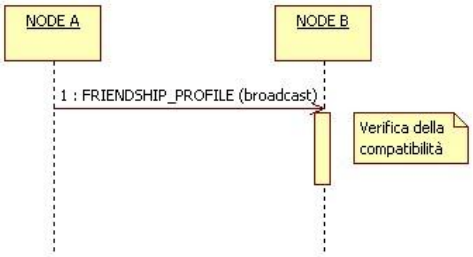

Figura 12 Profilo non compatibile

Questa struttura con doppia conferma è stata introdotta per mantenere il più possibile coerenti le relazioni di amicizia tra i nodi.
È tuttavia possibile che le liste dei diversi nodi non siano completamente coerenti, in quanto gli ack potrebbero andare perduti, ma si è scelto di ovviare a questa situazione in fase di chatting sincrono.
In questo protocollo abbiamo scelto di non prevedere ritrasmissioni, né per quanto riguarda l'invio della richiesta di amicizia per profilo, né per quanto riguarda la disponibilità di amicizia inviata dai nodi compatibili con il profilo richiesto. In entrambi i casi non è infatti possibile prevedere se si riceverà risposta, e quante risposte si riceveranno, perderebbe quindi di significato l'introduzione di una ritrasmissione. Si è scelto di lasciare risolvere l'eventuale perdita di messaggi dall'utente stesso, che in ogni momento può decidere di effettuare una nuova ricerca di amici per profilo.

4.3 Cancellazione di un amico

Un altro importante aspetto relativo alla gestione delle relazioni di amicizia è dato dalla possibilità di cancellare un amico dalla propria lista. Il messaggio di cancellazione di tipo FRIENDSHIP_DELETE, può essere inviato sia che l'amico sia presente (in questo caso verrà subito inviato un messaggio di rimozione), sia che non sia presente.
In caso l'amico sia offline il messaggio di cancellazione verrà trattato come i messaggi di conversazione offline.
In ogni caso la cancellazione comporta l'immediata eliminazione dell'amico dalla propria lista.

Figura 13 Form di rimozione di un amico

4.4 Aspetti implementativi

Per permettere lo svolgimento di questo servizio viene attivato su ogni nodo all'avvio dell'applicazione, un thread che si occupa di ricevere le richieste di amicizia e di inviare le risposte.
Questo thread si occuperà quindi di ricevere i messaggi FRIENDSHIP_UUID, FRIENDSHIP_PROFILE e FRIENDSHIP_DELETE (se il contatto che si vuole cancellare è presente al momento della cancellazione), e di inviare i messaggi FRIENDSHIP_ACK.

4.5 Test e risultati sperimentali

Sono stati effettuati alcuni test per verificare il corretto funzionamento del servizio implementato. Anche in questo caso, tali test sono stati eseguiti con solo sei dispositivi, di conseguenza prima di portare il servizio su più larga scala sarebbe opportuno testarlo con un numero maggiore di nodi connessi.
La perdita di messaggi è risultata piuttosto bassa, tale da non giustificare l'introduzione di ritrasmissioni.

Sono stati valutati:

- il tempo di attesa del primo ack successivo all'invio della richiesta di amicizia dalla parte del nodo che effettua la ricerca;
- il tempo di attesa tra l'invio di tale ack e la ricezione dell'ack finale di conferma dell'instaurazione dell'amicizia da parte del nodo che ha ricevuto la richiesta.

I tempi di ricezione dei messaggi (calcolati in ms) risultano piuttosto bassi:

Attesa primo ack in risposta alla richiesta di amicizia	Attesa ack finale di conferma dell'aggiunta dell'amico
1322	149
1362	128
1202	179

L'attesa dell'ack da parte del nodo che effettua la ricerca risulta maggiore in quanto prima dell'invio dell'ack viene chiesto all'utente se desidera stabilire una relazione di amicizia con l'utente da cui è stata ricevuta la richiesta. L'invio dell'ack finale invece non è dipendente da elaborazioni o interazioni con l'utente.

5 Sviluppi futuri

5.1 Replicazione

Sarebbe possibile introdurre in futuro una replicazione del Persistent Server. Potrebbe essere implementata sia una replicazione con elezione del sostituto, che darebbe maggiore qualità al servizio di messaggistica asincrona, sia una replicazione a copie attive del Persistent Server che potrebbe risolvere eventuali problemi di congestione che potrebbero verificarsi su larga scala.

5.2 Scalabilità

Sarebbe interessante analizzare le prestazioni e il funzionamento del sistema su larga scala, ma questo aspetto non è stato considerato in quanto il numero di dispositivi a nostra disposizione non permetteva questo tipo di analisi.
In caso di dense MANET è però un aspetto particolarmente significativo; potrebbe infatti essere necessario replicare alcuni servizi, o operare suddivisioni in sotto-MANET con la necessità di introdurre coordinazione tra le stesse.

5.3 Estensione al caso Multi-hop

Un ulteriore sviluppo futuro significativo è l'estensione al caso multi-hop. Questa estensione renderebbe necessaria l'introduzione di protocolli di discovery reattivi, che riducano l'overhead complessivo ma che riescano a recapitare messaggi anche a nodi non in diretta visibilità, tenendo traccia dei cammini e adattandoli dinamicamente alla MANET.

5.4 Servizi aggiuntivi

Si potrebbe inoltre pensare di estendere la gamma dei servizi offerti dall'applicazione, introducendo ad esempio la possibilità di gestire chat che prevedano la presenza di più partecipanti, o un servizio di invio file.

Bibliografia

[1] K. Topley, “J2ME in a nutshell”, O’Reilly, 2002.

[2] I. Chlamtac, M. Conti, JJN Li, “Mobile ad hoc networking: imperatives and challenger”, Elsevier Journal of Ad Hoc Networks, vol.1, no.1, 2003.

[3] A.S. Tanenbaum, M. van Steen, “Distributed Systems: principles and paradigms”, Prentice-Hall, 2002

[4] A. Corradi, Slides del Corso di Reti di Calcolatori L-S e Reti di Calcolatori L-A

[5] http://www.db4o.com

[6] https://www14.software.ibm.com

[7] http://www.sun.com