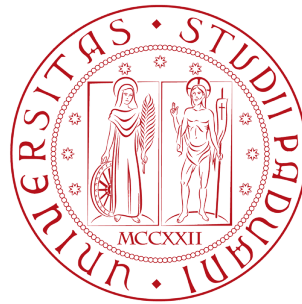


PARIIM: XMPP

RELATORE: Ch.mo Prof. Enoch Peserico Stecchini Negri De Salvi
CORRELATORE: Ing. Paolo Bertasi
LAUREANDO: Roberto Piva

A.A. 2009-2010



UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
TESI DI LAUREA

PariIM: XMPP

RELATORE: Ch.mo Prof. Enoch Peserico Stecchini Negri De
Salvi

CORRELATORE: Ing. Paolo Bertasi

LAUREANDO: *Roberto Piva*

A.A. 2009-2010

A chi ha creduto e crede in me
Al lettore interessato

Indice

Sommario	1
Introduzione	2
1 Il progetto PariPari	5
1.1 L'idea	5
1.2 Struttura interna	6
1.3 Il plugin IM e suo ruolo	7
2 Il protocollo XMPP	11
2.1 Introduzione	11
2.2 Funzionalità	13
2.2.1 Login da postazioni multiple	13
2.2.2 Comunicazione tra utenti di server diversi, o intercomuni- cazione tra server	14
2.2.3 Sicurezza e cifratura	14
2.2.4 Connessione da reti filtrate	15
2.2.5 Conversazioni a più utenti e conversazioni pubbliche	15
2.2.6 Supporto ai protocolli di terze parti	16
2.3 Chi lo usa (e come)	18
2.4 Motivazioni per adottare XMPP nel plugin IM	19
3 Il plugin IM, pianificazione del lavoro e sviluppo di XMPP	21
3.1 Struttura del plugin	21
3.1.1 ImConnectionManager	23
3.1.2 ImListener	23
3.1.3 Le interfacce	23

INDICE

3.2	Analisi di fattibilità	25
3.3	Libreria esterna: (a)Smack	26
3.3.1	Struttura di Smack	27
3.3.2	Problemi di Smack	28
3.3.3	Google Android e aSmack	29
3.4	Innesto in IM e interazione con altri plugin	29
4	Realizzazione	33
4.1	XmppUser	33
4.2	XmppContact / XmppContacts	37
4.3	XmppGroup	38
4.4	Difficoltà incontrate	39
4.5	Sviluppi futuri	40
4.5.1	Uso delle strutture socket del plugin Connectivity	40
4.5.2	Uso del sistema di messaggi interno di IM	40
4.5.3	Trasferimento file	41
4.5.4	Sfruttamento della funzionalità gateway di XMPP e sviluppo di un server dedicato	41
4.5.5	Un protocollo PariPariIM	41
5	Conclusioni	43
	Bibliografia	45

Sommario

La capillare diffusione delle connessioni a Internet è favorita dal crescente e autoalimentato desiderio di comunicazione insito nell'uomo. In particolare nella fascia di utenza più giovane si osserva come la comunicazione sia sempre più spesso veicolata attraverso strumenti elettronici, come gli SMS, o più recentemente email, chat e messaggistica istantanea. Questo elaborato vuole esporre i passi che mi hanno portato a realizzare (in parte) il supporto al protocollo XMPP¹ per il plugin IM del progetto PariPari. Il lavoro è stato svolto dall'ottobre 2009 al settembre 2010.

¹Extensible Messaging and Presence Protocol, vedi [8] e [9] e il capitolo 2

Introduzione

La nascita di Internet è stata dettata dalla necessità di garantire una comunicazione veloce e affidabile a prescindere dalla posizione degli interlocutori. Con l'evolversi della tecnologia e l'abbassamento dei prezzi i computer sono entrati nelle case di pressochè ogni persona, e successivamente il “bisogno” innato di informazione e comunicazione ha portato a le stesse persone a dotarsi di connessione a Internet. Oltre a WWW², sistema ormai universalmente conosciuto, un altro tipo di servizio molto utilizzato è quello di chat / messaggistica istantanea (da qui *im*³). Solitamente ci si riferisce a “chat” quando si parla di servizi tipo IRC⁴, mentre con “im” si intende un servizio di comunicazione diretta tra due computer, per lo scambio di brevi messaggi di testo.

Il servizio di messaggistica più noto e più utilizzato è Windows Live Messenger, ex MSN Messenger, che conta una bacino d'utenza di circa 300 milioni di utenti (dati giugno 2009⁵). Un altro servizio oggi sempre più importante e utilizzato è facebook, che vanta più di 500 milioni di utenti (dati facebook 5 novembre 2010⁶). Facebook è principalmente un social network, ma integra funzionalità di *im* (facebook chat [6]) con i propri “amici” attraverso una semplice interfaccia grafica non invasiva. Dal 2005 anche il servizio Google Mail di Google è dotato di funzionalità *im*, tale servizio si chiama Google Talk [7] (Gtalk da qui). Gtalk è integrato nell'interfaccia dell'applicazione web Google Mail con un'interfaccia analoga a quella della chat di facebook, inoltre è disponibile un client a sè stante, omonimo. I sistemi Gtalk e facebook chat sono accomunati dal protocollo che essi utilizzano per interfacciarsi agli utenti, tale protocollo è XMPP, ed è uno stan-

²World Wide Web, vedi [3]

³Instant Messaging, vedi [5]

⁴Internet Relay Chat, vedi [4]

⁵wikipedia: Instant Messaging [5], sezione [User base](#)

⁶<http://www.facebook.com/press/info.php?statistics>

dard internazionale IETF⁷, Windows Live Messenger invece utilizza un protocollo proprietario, denominato MSNP⁸.

La piattaforma PariPari [1], ambizioso progetto *peer to peer* multifunzionale, è dotata di un plugin, chiamato “IM”, che ha come obiettivo dotare PariPari di funzionalità, appunto, di instant messaging⁹, fornendo il supporto ai più comuni protocolli di comunicazione. Attualmente è presente un supporto parziale al protocollo MSNP, questo elaborato descriverà invece il processo di realizzazione, anch’essa parziale, del supporto al protocollo XMPP per il plugin IM.

Nel primo capitolo verrà introdotto il progetto PariPari e la sua struttura, a seguire posizione e ruolo di IM. Il secondo capitolo spiegherà brevemente il protocollo XMPP e le sue potenzialità, corredando la trattazione con casi d’uso reali. Nel terzo capitolo verrà spiegato il processo di pianificazione dello sviluppo del supporto a XMPP, dall’analisi di fattibilità alle scelte operate. Il quarto capitolo tratterà la realizzazione del lavoro, ed esporrà qualche spunto sulle possibili funzionalità da sviluppare in futuro. Per finire le conclusioni di riepilogo sul lavoro svolto.

⁷Internet Engineering Task Force, <http://www.ietf.org/>

⁸Microsoft Notification Protocol

⁹da qui con *im* si intenderà il servizio di instant messaging, mentre con “IM” il plugin di PariPari

Capitolo 1

Il progetto PariPari



Figura 1.1: Il logo di PariPari

PariPari¹ è una rete peer-to-peer (da qui p2p) pura basata su una tabella di hash distribuita simile a Kademia². Il progetto ha come obiettivo quello di garantire l'anonimato degli utenti, di creare un sistema intelligente di crediti e fornire una rete multifunzionale.

1.1 L'idea

L'aspetto più innovativo e al contempo ambizioso di PariPari è la multifunzionalità: l'idea di fondo è di creare una rete nella quale siano disponibili tutti i più comuni servizi offerti dalla Internet. Tali servizi saranno costruiti in modo da essere distribuiti sulla rete e disponibili anche all'esterno di PariPari stessa. Questo approccio di distribuzione garantirebbe la raggiungibilità del servizio anche qualora un nodo della rete venga chiuso. Si pensi ad esempio ad un server web

¹vedi [1]

²vedi [11]

distribuito tra molti nodi della rete: anche nel caso uno o più nodi vengano chiusi esso sarebbe comunque disponibile nella rete, in quanto i nodi rimanenti potranno rigenerare la parte mancante. Una volta che tutti i servizi più importanti saranno distribuiti sulla rete PariPari essa potrà diventare, di fatto, una internet dentro Internet (attenzione alle maiuscole), ma grazie alla distribuzione sarà più stabile e resistente alla caduta dei singoli nodi.

Un ulteriore vantaggio dell'approccio p2p è la scalabilità: dato che ogni nodo si occupa di contribuire ad autosostenere la rete non sarà necessario investire in potenza dei nodi per evitare il collasso della rete stessa all'aumentare del numero di nodi collegati..

Per perseguire gli obiettivi del progetto occorre che un gran numero di persone adotti PariPari, questo perchè maggiori sono gli utenti maggiori sono i benefici che ciascuno potrà avere dall'utilizzare la rete³. Per ottenere questo è necessario fornire agli utenti, almeno inizialmente, delle funzionalità “interessanti” o “di moda”, come ad esempio la messaggistica istantanea o il download dalle più utilizzate reti di condivisione file p2p. PariPari farà anche questo per mirare ad essere l'applicazione p2p più utilizzata nel mondo.

1.2 **Struttura interna**

La struttura di PariPari è modulare, ogni modulo è detto plugin. Un plugin è un programma non autonomo che sviluppa funzionalità utili o aggiuntive per il programma principale, con il quale interagisce. In PariPari il programma principale è Core, esso si occupa di gestire le richieste e di assegnare le risorse disponibili in modo da evitare lo stallo dei plugin. Core fa parte della così detta “cerchia interna”, ovvero fa parte dei plugin che offrono le funzionalità di base per l'accesso al sistema sottostante e alla connettività di rete e p2p. Della cerchia interna fanno parte, oltre a Core:

Crediti per gestire i crediti della rete p2p ⁴

³il così detto “effetto rete”, vedi http://en.wikipedia.org/w/index.php?title=Network_effect&oldid=394297072

⁴vedi ad esempio la rete ED2K su http://en.wikipedia.org/wiki/EDonkey_network, e su www.emule-project.net

Connectivity fornisce *socket* e altre forme di connettività via rete agli altri plugin

Storage offre modalità di lettura e scrittura di file.

DHT fornisce le funzionalità di una tabella di hash distribuita, utile per la gestione della rete p2p

Oltre alla cerchia interna ci sono i plugin che sviluppano le funzionalità che un utente poi utilizzerà. Tali plugin fanno parte della così detta “cerchia esterna” e sono, tra gli altri:

Mulo/Torrent client delle reti p2p ED2K e torrent⁵

IRC per l’accesso alle chat con protocollo IRC

IM per la messaggistica istantanea, multiprotocollo

Al momento della scrittura di questo elaborato non è disponibile un plugin completo e funzionante che si occupi di creare un’interfaccia grafica per PariPari. Tale plugin è attualmente in sviluppo e avrà un ruolo chiave per la futura diffusione di PariPari, dato che sarà la prima cosa che i nuovi utenti vedranno ed è necessario rendere l’approccio il più semplice ed immediato possibile.

1.3 Il plugin IM e suo ruolo

PariPari nasce con l’obiettivo di fornire una piattaforma centrale per l’utenza per le più disparate modalità di condivisione nella Internet. In questo senso PariPari, in una tipica sessione di lavoro, dovrebbe consentire di svolgere la maggiore quantità possibile di interazioni-tipo tra l’utente e la sua rete sociale, in tutte le accezioni possibili. L’utente infatti è legato al concetto di rete a diversi gradi di coscienza della stessa. In particolare, l’utenza media di una rete di file sharing è interessata, di fatto, al mero ottenimento della risorsa ricercata, poco importa chi siano i condivisori effettivi. Al contrario l’utente di un servizio di *im* è fortemente interessato agli individui con cui si relaziona, perchè è più vicino alla sua realtà di interazione sociale.

⁵www.bittorrent.com

1. IL PROGETTO PARIPARI

Quest'ultima è, oggi, la modalità principe di interazione sociale. Celeberrimo è l'esempio di facebook, il primato di social network (rete sociale, appunto) più frequentato al mondo, che può contare su una base d'utenza di circa 500 milioni di individui. Per poter comprendere appieno la rilevanza di questo dato basti confrontarlo con le dimensioni stimate della rete ED2K, che comprende un numero di utenti dell'ordine dell'unità dei milioni. Nell'ottica di un futuro lancio di PariPari nel mercato occorre fare in modo che la piattaforma sia appetibile alla nuova utenza. Una strategia vincente è l'integrazione dei servizi oggi più utilizzati in un unico luogo virtuale. PariPari non può esularsi dal distinguersi nel mercato offrendo servizi nettamente migliori della concorrenza. IM è la soluzione a tale problema, esso fornisce la funzionalità di instant messaging alla piattaforma, sfruttando le reti concorrenti. Grazie a IM, quindi, PariPari diventa una piattaforma con la quale si preserva la connessione alle reti precedentemente utilizzate e ne si centralizza l'interfaccia, in maniera totalmente trasparente all'utente. Ottenuto questo risultato gli utenti delle altre reti avranno solo vantaggi utilizzando PariPari, in quanto offre sia i servizi di interazione sociale sia servizi accessori, il tutto con una spiccata propensione all'interazione tra servizi di diversa natura.

Sotto questa luce occorre analizzare quali siano i servizi *im* più utilizzati tra quelli attualmente disponibili, con lo scopo di spingere il maggior numero possibile di persone a migrare verso PariPari sfruttandone la maggiore base d'utenza. Questo approccio è volto a minimizzare il tempo necessario a PariPari per raggiungere la soglia minima di utilità grazie all'effetto rete. Oltre al già citato facebook abbiamo Windows Live Messenger, con 330 milioni di utenti attivi nel 2009, Skype, il servizio di VOIP⁶ p2p più utilizzato al mondo, con 309 milioni di utenti nel 2008 e Yahoo Messenger, che conta circa 250 milioni di utenti nel 2008. Altri servizi, come AIM e ICQ, vantano una base d'utenza ampia in sé ma molto piccola rispetto a quelle precedentemente citate. Discorso a parte per la rete di Tencent QQ, che con la sua utenza di 440 milioni di utenti attivi su 990 milioni totali vincerebbe il confronto con le altre reti, ma è tuttavia utilizzato solo in Cina. Da citare anche il servizio Gmail di Google, che offre, parallelamente alla posta elettronica, un servizio di *im* chiamato Gtalk, integrato nell'interfaccia⁷. Un confronto grafico tra gli utilizzi dei vari protocolli è visibile nella figura 1.2.

⁶Voice Over IP

⁷Gmail vanta una media di 193 milioni di utenti mensili, dati 2010, vedi <http://en.wikipedia.org/wiki/Gmail>

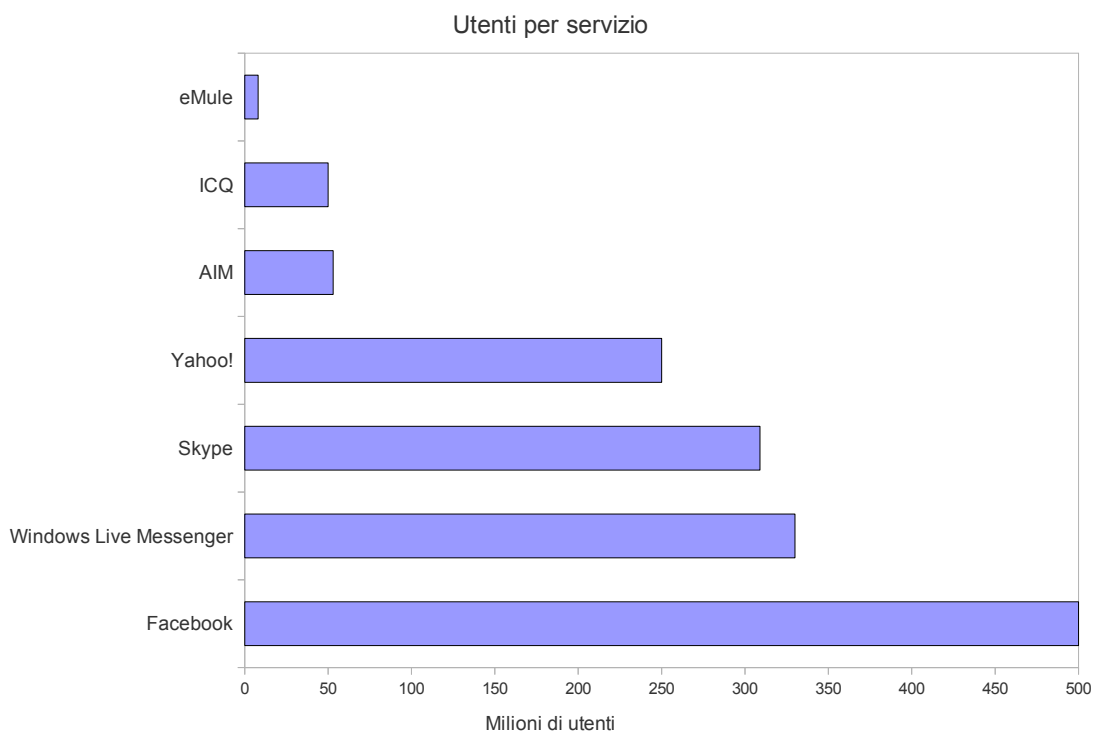


Figura 1.2: Statistiche d'uso dei principali servizi di chat, in confronto con gli utenti di ED2K/eMule

Dal febbraio 2010 facebook offre la possibilità di connettersi al proprio servizio di *im* interno attraverso il protocollo XMPP (vedi 2). Tale protocollo è, incidentalmente, sfruttato anche dal servizio Gtalk, si è quindi pensato di includere il supporto a tale protocollo nel plugin IM al fine di spingere l'utenza dei due servizi a migrare verso PariPari.

1. *IL PROGETTO PARIPARI*

Capitolo 2

Il protocollo XMPP

2.1 Introduzione

Il protocollo XMPP, o Extensible Messaging and Presence Protocol, nasce dall'esigenza di un protocollo di comunicazione in tempo reale aperto, sicuro e decentralizzato, alternativo ai servizi di *im* chiusi al tempo disponibili. Nel 1999 la comunità open source Jabber comunicò l'esistenza di Jabber, una tecnologia aperta di messaggistica e notificazione di presenza, che in seguito cambierà nome in XMPP. Nel 2002 IETF¹ formò il XMPP Working Group con lo scopo di formalizzare lo standard per XMPP. Tale gruppo produsse 4 RFC² che verranno poi approvati da IETF nel 2004 trasformando quindi XMPP in uno standard riconosciuto. XMPP è mantenuto e aggiornato dalla XMPP Standards Foundation (XSF, ex JSF³).

XMPP offre diversi vantaggi rispetto ai protocolli proprietari concorrenti:

Apertura il protocollo è aperto, libero, gratuito e pubblico. Ogni persona che lo desidera può scrivere la propria implementazione, sia dal lato server sia dal lato client, e può produrre ovviamente delle librerie di supporto.

Standardizzazione il protocollo è uno standard IETF basato su *stream XML*⁴.

Gli RFC di riferimento sono RFC 3920 (core) e RFC 3921 (estensioni al

¹Internet Engineering Task Force, <http://www.ietf.org/>

²Request for Comments, vedi <http://www.ietf.org/rfc.html>

³XMPP Standards Foundation e Jabber Standards Foundation

⁴Extensible Markup Language, vedi [12]

2. IL PROTOCOLLO XMPP

protocollo), RFC 3922 (mappatura di XMPP su Common Presence and Instant Messaging, CPIM) e RFC 3923 (cifratura end-to-end).

Decentralizzazione grazie all'architettura del protocollo ogni persona può realizzare e/o avviare un proprio server XMPP, mantenendo al contempo la possibilità di intercomunicare con gli altri server della rete.

Sicurezza i server possono essere isolati dalla rete pubblica, XMPP poi offre la possibilità di cifrare le connessioni, grazie alle tecnologie SASL⁵ e TLS⁶, già dalla definizione del core. Inoltre è sempre possibile cifrare il testo dei messaggi al fine di garantire la sicurezza anche in caso di *stream* in chiaro: tale pratica è chiamata Off-the-Record Communication⁷.

Estensibilità grazie a XML è possibile definire nuove funzionalità per XMPP, ad esempio fornendo il servizio di chat vocale attraverso un protocollo di VOIP⁸. Per mantenere l'interoperabilità la XFS si occupa di tracciare le estensioni popolari (l'indirizzo a cui reperirne una descrizione è <http://xmpp.org/xmpp-protocols/xmpp-extensions/>), tuttavia non è richiesta la pubblicazione e pertanto le organizzazioni possono sviluppare ed utilizzare estensioni private se desiderato.

Flessibilità XMPP, grazie alla sua natura basata sui messaggi, si presta a risolvere compiti non strettamente legati all'instant messaging. Esistono applicazioni basate su XMPP che si occupano di controllo delle reti, distribuzione di contenuti, scambio di file, strumenti di collaborazione remota, servizi web, strumenti di *middleware* e molto altro.

Lo standard XMPP definisce inoltre una porta TCP di riferimento per la connessione dei client al server (la 5222) e per la comunicazione tra server (la 5269), entrambe registrate alla IANA⁹

⁵Simple Authentication and Security Layer

⁶Transport Layer Security

⁷vedi <http://www.cypherpunks.ca/otr/otr-wpes.pdf>

⁸Voice Over IP

⁹Internet Assigned Numbers Authority, www.iana.org

2.2 Funzionalità

XMPP nasce anche con l'obiettivo di fornire alcune funzionalità aggiuntive interessanti rispetto alla concorrenza. In questa sezione vedremo alcune caratteristiche poco comuni o addirittura peculiari di XMPP.

2.2.1 Login da postazioni multiple

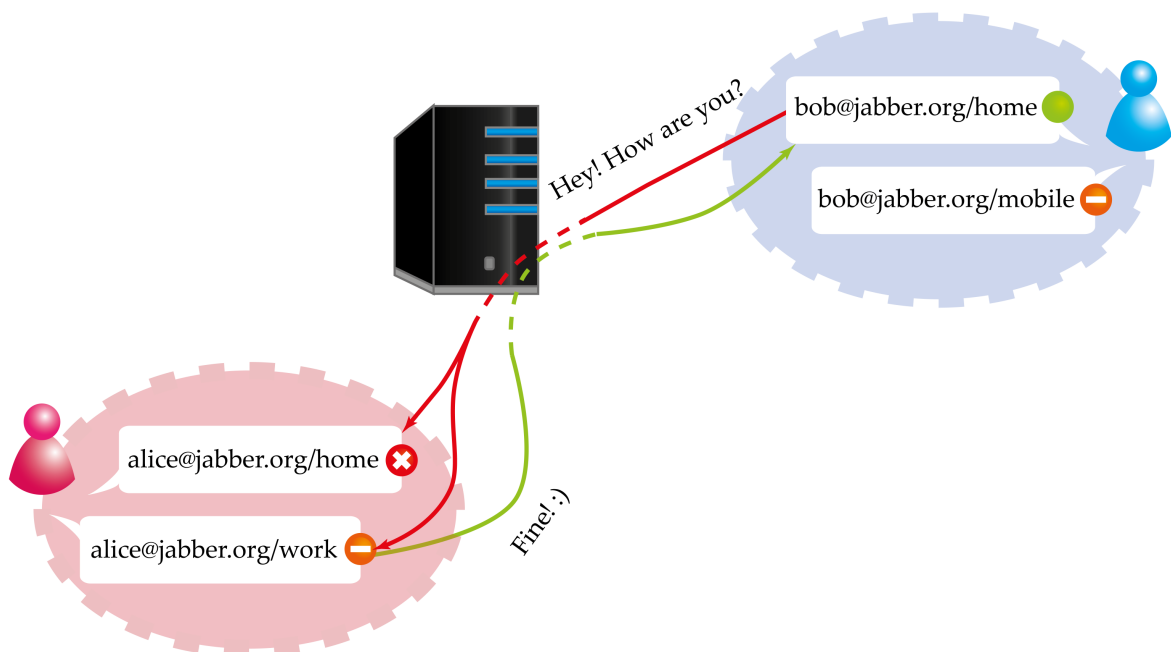


Figura 2.1: Schema di conversazione tra utenti connessi da postazioni multiple

XMPP permette l'autenticazione da postazioni multiple già dalla prima bozza di RFC. L'utente ha la possibilità di effettuare il login al proprio server (ad esempio `alice@jabber.org` sul server `jabber.org`) fornendo un identificativo di "risorsa". Tale risorsa, assieme al nome utente e al dominio, costituisce il JID, o Jabber ID, e va scritto nella forma `alice@jabber.org/risorsa`. Il server di riferimento al quale collegarsi per il login può differire dalla parte di JID dopo il simbolo "@", ad esempio il server per gli utenti Gtalk (con JID del tipo `nomeutente@gmail.com`) è `talk.google.com` anziché `gmail.com`. Ogni risorsa può avere il proprio stato, ad esempio Alice può lasciare il computer di casa acceso e connesso al server jabber con il JID `alice@jabber.org/casa` e

stato “Non al computer”, e al contempo accedere con il suo *smartphone* con JID `alice@jabber.org/mobile` e stato “Disponibile”. Questo permette a client specializzati di comunicare ad una specifica risorsa di un contatto, se necessario. Se lo scopo è invece cercare il contatto ovunque esso sia disponibile, è sufficiente spedire il messaggio al semplice `nomeutente@dominio` senza menzionare la parte risorsa, e il server lo spedirà a tutte le risorse per quel contatto. Il client, ottenuta la risposta, si occuperà di spedire i successivi messaggi alla risorsa che risponderà per prima.

Per correttezza bisogna citare MSNP, che dalla versione 16 offre un servizio simile, chiamato MPOP¹⁰, nato con lo scopo di permettere login multipli anche per Windows Live Messenger.

2.2.2 Comunicazione tra utenti di server diversi, o intercomunicazione tra server

I server XMPP costituiscono una rete simile a quella del servizio mail, ogni server, se configurato opportunamente, può inoltrare messaggi diretti ad utenze non sue. Ad esempio se Alice ha un account su `jabber.org` e Bob su `linuxlovers.at` Alice può comunicare con Bob spedendo a `jabber.org` un messaggio indirizzato a `bob@linuxlovers.at`, e il suo server si occuperà di inoltrare il messaggio al server di `linuxlovers.at` demandandone la consegna a Bob. Le comunicazioni tra server avvengono a loro volta tramite il protocollo XMPP, ma su una porta diversa, la TCP 5269.

2.2.3 Sicurezza e cifratura

XMPP offre la possibilità di autenticare gli utenti in maniera sicura utilizzando il framework SASL¹¹. Questo protocollo è spesso utilizzato in concomitanza con il protocollo TLS¹² per la cifratura dell'intera connessione. La combinazione dei due garantisce un alto grado di riservatezza durante la conversazione. Occorre precisare che se il destinatario di una comunicazione utilizza un sistema non protetto è ancora possibile intercettare le conversazioni.

¹⁰Multiple Points Of Presence

¹¹Simple Authentication and Security Layer, <http://tools.ietf.org/html/rfc4422>

¹²Transport Layer Security, <http://tools.ietf.org/html/rfc5246>

2.2.4 Connessione da reti filtrate

Uno dei problemi che affligge alcuni protocolli di *im* è l'impossibilità di fruire il servizio da reti protette da firewall. In tali reti vengono spesso impedito le connessioni verso l'esterno su porte non conosciute o su porte relative a servizi di *im*. Nelle situazioni peggiori le uniche porte di uscita che gli utenti sono autorizzati a contattare sono quelle relative ai servizi HTTP¹³ (TCP 80) e HTTPS¹⁴ (TCP 443), ovvero quelle preposte alla navigazione su Internet. Con delle estensioni a XMPP è possibile veicolare il traffico XML, altrimenti bloccato, attraverso il protocollo HTTP su porta TCP 80, eludendo i filtri della rete in cui si risiede. Tale risultato è ottenuto utilizzando un meccanismo a *polling*¹⁵ oppure a *binding*¹⁶. Il polling, oggi obsoleto, consiste nell'interrogare periodicamente il proprio server, attraverso richieste HTTP GET o POST¹⁷, sulla presenza o meno di nuovi messaggi. Con il binding il client sfrutta il meccanismo di *keep-alive* di HTTP¹⁸ per mantenere la connessione col server attiva; in questo modo il server può inviare spontaneamente i nuovi messaggi al client appena essi arrivano, sfruttando la connessione precedentemente instaurata. Questo secondo meccanismo garantisce una maggiore efficienza in confronto al polling, dove la maggior parte dei messaggi ha contenuto nullo.

2.2.5 Conversazioni a più utenti e conversazioni pubbliche

XMPP supporta nativamente le chat di gruppo, vedi XEP-0045¹⁹, sul modello di IRC. Le conversazioni multiple vengono organizzate in "stanze", ognuna delle quali può essere raggiunta da più utenti. Ogni stanza viene registrata sul server come `nomestanza@servizio`, dove con "nomestanza" si intende un nome identificativo, mentre con "servizio" si intende il nome di dominio del servizio conferenza del server scelto, ad esempio `conference.jabber.org`. Gli utenti del servizio conferenza possono creare nuove stanze, specificando alcune caratteristiche di queste, tra cui:

¹³Hypertext Transfer Protocol

¹⁴HTTP over Secure Socket Layer

¹⁵<http://xmpp.org/extensions/xep-0025.html>

¹⁶<http://xmpp.org/extensions/xep-0124.html>

¹⁷vedi http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

¹⁸vedi HTTP e <http://en.wikipedia.org/wiki/Keepalive>

¹⁹<http://xmpp.org/extensions/xep-0045.html>

- Scegliere l'oggetto della discussione
- Dichiarare la stanza pubblica o privata, ovvero se chiunque può interrogare il servizio per venire a conoscenza della presenza di tale stanza
- Dichiarare una stanza come persistente, ovvero mantenuta “viva” anche dopo la disconnessione o l'allontanamento di tutti i suoi partecipanti
- Apporre una password per l'ingresso alla stanza
- Imporre restrizioni di accesso, ovvero permettere o meno l'ingresso di persone qualunque nella stanza, oppure limitare l'accesso ai soli invitati.
- Dichiarare alcuni utenti come moderatori di tale stanza, consentendo loro di allontanare altri utenti dalla stessa.
- Rendere gli utenti della stanza anonimi o semi-anonimi

Una volta elencate queste proprietà è evidente che XMPP si propone come alternativa sia ai sistemi di instant messaging puri sia a servizi come IRC, unendo le funzionalità di entrambi e aggiungendo nuove funzioni alla somma delle precedenti.

2.2.6 Supporto ai protocolli di terze parti

Il protocollo XMPP supporta una particolare funzione: il *gateway*²⁰ (a volte chiamato *transport*). Un gateway è un server che fornisce l'accesso a reti di terze parti ai server XMPP. Tale servizio può essere offerto dalla stessa macchina fisica in cui risiede il server XMPP oppure in un'altra locazione. Il gateway consente ai client di connettersi a protocolli di terze parti senza aver bisogno di un supporto apposito alle reti desiderate. Il client dovrà notificare al proprio server l'intenzione di servirsi di un gateway per un certo protocollo di terze parti. Successivamente fornirà al proprio server le credenziali d'accesso al protocollo di terze parti al proprio server affinché le spedisca al gateway. Quest'ultimo provvederà a fornire un elenco di contatti in formato XMPP al proprio server, e quindi l'utente vedrà nella propria lista contatti anche quelli della rete extra-XMPP.

²⁰<http://xmpp.org/extensions/xep-0100.html>

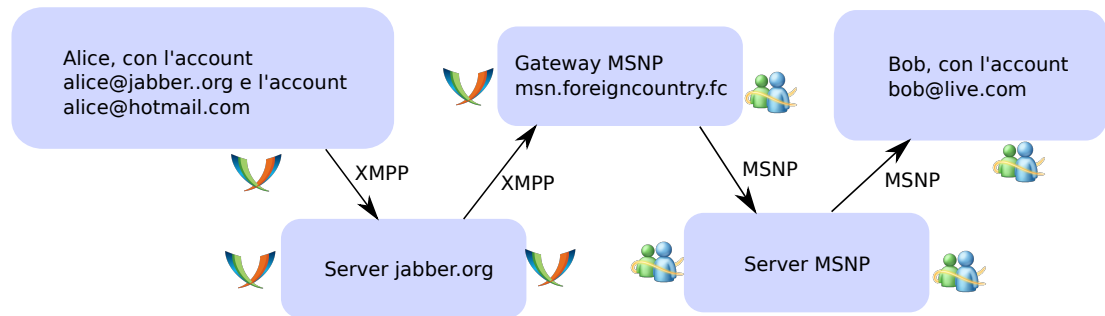


Figura 2.2: Esempio sulla funzionalità di gateway

Vediamo un esempio: con riferimento alla figura 2.2 poniamo che Alice abbia un account su `jabber.org` e voglia usufruire di un servizio di connessione al protocollo MSNP locato all'indirizzo `msn.foreigncountry.fc`. Alice fornirà al proprio server le proprie credenziali del servizio Windows Live Messenger, unitamente all'indirizzo `msn.foreigncountry.fc`, al proprio server. Il server `jabber.org` comunicherà a `msn.foreigncountry.fc` le credenziali di Alice e ritornerà ad essa un elenco di contatti del tipo `bob.live.com@msn.foreigncountry.fc`²¹, `othercontact.hotmail.com@msn.foreigncountry.fc`, eccetera, che sono la rappresentazione in JID dei contatti di Alice sulla rete Windows Live Messenger. Per ottenere questo Alice necessita solo di un client compatibile con XMPP.

Attualmente esistono molti gateway, tra i principali vi sono:

AIM Per il protocollo AOL Instant Messenger.

ICQ Per il protocollo omonimo.

MSN Per i protocolli Microsoft Window Live Messenger.

Yahoo! Per il servizio Yahoo! Messenger.

SMS Per un servizio di spedizione SMS²², ove possibile.

Email Per un servizio di SMTP²³ gateway.

IRC Per il collegamento a server irc.

²¹è evidente che il simbolo "@" non possa comparire nella parte "nomecontatto" in quanto violerebbe la forma dei JID; ogni gateway ha la sua convenzione per "tradurre" il simbolo in un altro lecito, in questo caso "."

²²Short Message System


²³Simple Mail Transfer Protocol

Gtalk Per il collegamento al servizio Gtalk, anche se ad esso si può accedere direttamente tramite protocollo XMPP.

Una lista di server con relativi servizi gateway offerti si può trovare all'indirizzo <http://www.jabberes.org/servers/>.

2.3 Chi lo usa (e come)

Il protocollo XMPP è molto diffuso, sul sito ufficiale del protocollo le ultime notizie parlano di circa 10 milioni di utenti sparsi tra le centinaia di migliaia di server²⁴. E' poco noto tuttavia che sono molte le compagnie che utilizzano XMPP per i loro prodotti o che espongono un'interfaccia XMPP agli utenti di questi. In questa sezione citiamo alcuni esempi:

 Google utilizza XMPP in varie applicazioni, in particolare il servizio di *im* Google Talk²⁵ (Gtalk) è basato sul servizio XMPP ed inoltre consente di aggiungere contatti di altri server, grazie al servizio *dialback*²⁶. Gtalk è inoltre integrato nell'interfaccia Gmail, nel qual caso verrà utilizzato il collegamento al server XMPP mediante HTTP.

La piattaforma Google Wave²⁷, una sorta di connubio tra *im*, email, forum e strumento di collaborazione remota, utilizza, nella comunicazione tra client-server e server-to-server, un'estensione del protocollo XMPP²⁸.

facebook

Facebook, dal febbraio 2010, offre un'interfaccia XMPP compatibile per la connessione di client esterni al servizio di *im* interno al social network.

Purtroppo il servizio è limitato al solo instant messaging, non sono consentite le operazioni di aggiunta e rimozione contatti, la connessione a servizi di gateway e tutte le altre funzionalità offerte dal protocollo XMPP.

²⁴vedi <http://xmpp.org/xsf/press/>

²⁵vedi http://en.wikipedia.org/wiki/Google_Talk e <http://www.google.com/talk/>

²⁶vedi <http://www.ietf.org/rfc/rfc3920.txt>, sezione "Server Dialback"

²⁷vedi wave.google.com

²⁸vedi <http://www.waveprotocol.org/> e <http://www.waveprotocol.org/protocol/draft-protocol-specs/draft-protocol-spec>

NOKIA Nokia utilizza XMPP come protocollo per il suo Ovi Contacts. Come specificato nel sito ufficiale²⁹ l'obiettivo di Nokia è rendere disponibile una piattaforma di instant messaging versatile, che consenta ai possessori di *smartphone* della casa finlandese di mantenersi in contatto anche con utenti di altre reti compatibili XMPP, come Gtalk, utilizzando il client Nokia Ovi.

2.4 Motivazioni per adottare XMPP nel plugin IM

Dopo aver visto quali aziende adottano XMPP, alla luce di quanto detto nella sezione 1.3, e tenendo conto della base d'utenza dei servizi citati possiamo affermare che l'adozione di XMPP porterà un numero spropositato di utenti a valutare la migrazione verso la piattaforma PariPari. Non bisogna inoltre trascurare una considerazione implementativa: grazie alla funzionalità dei gateway, descritta nella sezione 2.2.6, sarà possibile, una volta sviluppato il supporto completo a XMPP, usufruire dei suddetti per ampliare il numero di protocolli supportati dal plugin. Tale approccio può costituire una soluzione temporanea nel caso si pensi di sviluppare in futuro il supporto ad altri protocolli in maniera nativa in IM, oppure come soluzione definitiva se si optasse per lo sviluppare un server XMPP distribuito su PariPari³⁰.

²⁹ <http://betalabs.nokia.com/apps/ovi-contacts>

³⁰Per una discussione più approfondita su queste ultime affermazioni rimandiamo alla sezione 4.5

2. *IL PROTOCOLLO XMPP*

Capitolo 3

Il plugin IM, pianificazione del lavoro e sviluppo di XMPP

Il plugin IM è un client multiprotocollo di messaggistica istantanea, sul modello di Pidgin¹. Il suo obiettivo è fornire l'accesso trasparente a diversi protocolli di messaggistica. In questo modo, attraverso un'interfaccia comune, l'utente sarà "ignaro" del protocollo che al momento sta utilizzando per comunicare col suo contatto, potendosi quindi concentrare unicamente nella conversazione.

3.1 Struttura del plugin

Per creare un client multiprotocollo il primo passo è definire una struttura comune per tutti i protocolli di *im*. Questo approccio consente di svolgere le funzioni comuni con un'unico pezzo di programma, lasciando ai client specifici il compito di implementare le funzioni specifiche e i dettagli di protocollo [2].

Il plugin IM ha il suo centro in due classi: `ImConnectionManager` e `ImListener`. Tali oggetti sono, di fatto, dei centri di smistamento stringhe. Ogni stringa rappresenta un pezzo di messaggio, corredata delle informazioni riguardanti mittente, destinatario e protocollo. Le stringhe sono raccolte da `ImListener` e vengono instradate da `ImConnectionManager` (vedi figura 3.1).

¹<http://pidgin.im/>

3. IL PLUGIN IM, PIANIFICAZIONE DEL LAVORO E SVILUPPO DI XMPP

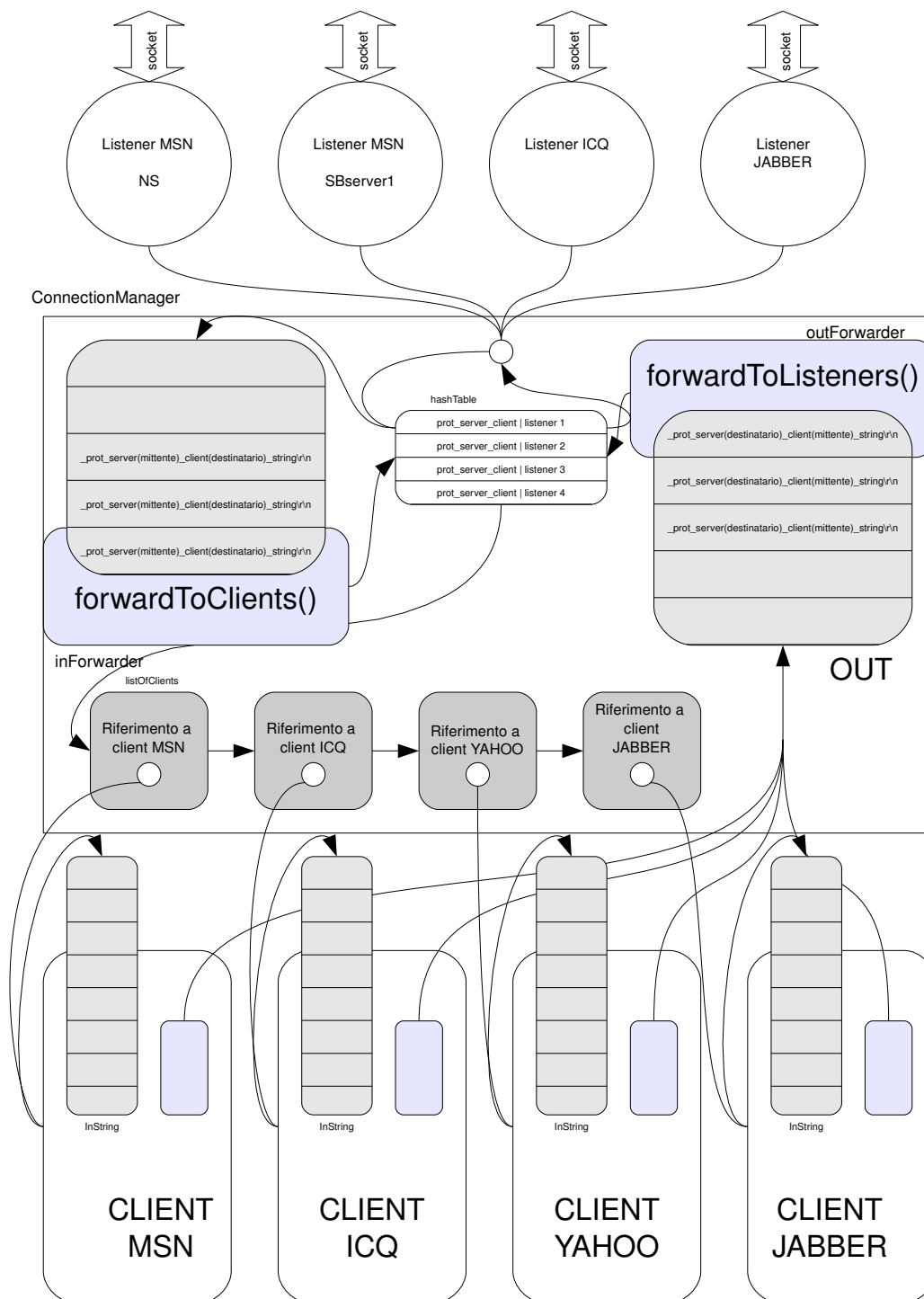


Figura 3.1: Schema del connection manager del plugin IM, G. Vallini [2]

3.1.1 ImConnectionManager

ImConnectionManager si occupa, come dice il nome, di gestire le connessioni necessarie al plugin. Oltre a questo svolge altre funzioni tra cui:

- Creazione e distruzione dei client dei singoli protocolli su richiesta (creazione alla connessione e distruzione alla disconnessione)
- Memorizzazione dei client connessi in un'apposita struttura
- Gestione delle connessioni con l'esterno (richiesta e rinnovo dei socket al plugin Core, apertura e chiusura connessioni, tracciamento delle connessioni attive)
- Gestione dell'inoltro di messaggi dall'esterno verso i client e viceversa.

L'inoltro da e verso i client è gestito tramite due code comuni, inQueue e outQueue, che operano in concorrenza. La coda inQueue riceve i messaggi dall'esterno attraverso i client e viene svuotata dal thread inForwarder. La coda outQueue riceve i messaggi dai vari client diretti all'esterno e viene svuotata dal thread outForwarder. I thread si occupano di leggere i messaggi e inoltrarli in base al destinatario.

3.1.2 ImListener

L'oggetto ImListener è deputato alla ricezione e invio di messaggi su socket. Ad ogni socket istanziato da ImConnectionManager viene associata un'istanza di ImListener, che scriverà i messaggi in uscita sul socket e rimarrà in attesa dei messaggi in arrivo. La creazione e distruzione dei Listener è contestuale alla creazione e distruzione di ogni socket. ImConnectionManager si occupa di mantenere la relazione tra ImListener e destinatari o mittenti dei messaggi.

3.1.3 Le interfacce

Per facilitare il lavoro di creazione del supporto ai vari protocolli sono state definite delle interfacce comuni. Sono state definite interfacce per i client dei singoli protocolli, gli utenti di un protocollo, i contatti di un utente e le liste di contatti:

IClient: rappresenta un'astrazione di un client per un protocollo. Per ogni protocollo vi è una classe che realizza l'interfaccia IClient che si occupa di

3. IL PLUGIN IM, PIANIFICAZIONE DEL LAVORO E SVILUPPO DI XMPP

gestire tutte le utenze di quel protocollo. In questo senso `IClient` è un “contenitore di utenti”, viene cioè istanziato una volta (e una sola volta) e poi gestisce tutti e soli gli utenti di quel protocollo (e solo di quello). `IClient` descrive alcune funzioni comuni utilizzate dalla classe che lo istanzierà (`ImConnectionManager`):

- Aggiungere o togliere un utente alla lista di utenti.
- Inserire un messaggio nella lista di input (il client si occuperà di smistarla al corretto utente).
- Trovare un utente di quel protocollo e ottenerne l’istanza della classe `IUser`.
- Ottenere una lista di utenti.

`IContact`: è una generica astrazione di contatto, contiene dei dati generici (email, *nickname*), delle rappresentazioni dello stato e descrive dei metodi per:

- Leggere e impostare lo stato del contatto.
- Leggere e modificare i dati generici.
- Bloccare il contatto.
- Determinare se il contatto è online o no.

`IUser`: è un’astrazione dell’utente di un protocollo, estende `IContact` e ne amplia le funzionalità con i metodi per:

- Ottenere la lista dei contatti
- Impostare la password d’accesso
- Ottenere la lista dei contatti online, bloccati eccetera (lista dei contatti filtrata)

`IContacts`: è la rappresentazione della lista contatti di un utente. Essa prevede modalità per:

- Aggiungere contatti
- Ricercare contatti
- Verificare se un contatto è online dato il suo identificativo (email)

Tali interfacce hanno permesso un alto livello di astrazione in fase di realizzazione del corpo centrale del plugin IM, consentendo di scriverlo una volta sola per tutti i protocolli, bug permettendo.

3.2 Analisi di fattibilità

Le potenzialità di XMPP lo rendono completo e versatile. La sua natura basata su XML e la standardizzazione rendono relativamente semplice la scrittura di client per tale servizio. Bisogna tuttavia considerare la vastità di casi d'uso che si devono affrontare nel corso della connessione. Il prezzo da pagare per l'alta versatilità è l'eccesso di opzioni di configurazione e di risposte ottenibili dal server: vi sono una vastità di errori da gestire, opzioni altamente specifiche da impostare nei pacchetti in uscita, funzionalità definite come estensioni ma da considerare come parte integrante del protocollo. Tutto questo fa lievitare la quantità di codice necessaria alla realizzazione del client. A questo proposito vanno fatte alcune considerazioni sulla correttezza e sull'efficienza:

- Scrivere grandi quantità di codice porta inevitabilmente a introdurre degli errori, occorre molta attenzione ed esperienza per evitare questo.
- La quantità di funzionalità da scrivere è grande anche se si volesse realizzare un client minimale
- Si deve garantire che il client funzioni bene nel maggior numero di casi possibile, in relazione ai concetti menzionati in 2.2.4. Senza questa caratteristica l'utenza, scoraggiata dal mancato funzionamento del comparto *im*, tenderà ad abbandonare PariPari, anche qualora offrisse altri servizi di grande qualità (vedi 1.3)
- Il codice deve essere il più efficiente possibile, compatibilmente con il tempo a disposizione per la stesura. Un client poco "reattivo" è comparabile a un client non funzionante, portando alle stesse conseguenze.
- Sul progetto PariPari, e quindi anche su IM, lavorano persone in formazione, non programmatori professionisti (almeno non necessariamente), è perciò ardito supporre a priori che il codice prodotto sia corretto, completo ed efficiente allo stesso tempo.

- La considerazione precedente va sommata al fatto che i programmatori di PariPari sono studenti. Questo sta a significare che non tutto il loro tempo sarà votato alla programmazione, al contrario dei professionisti, pertanto non è possibile stimare precisamente il tempo necessario al completamento del client.

Fatte queste considerazioni si è optato per cercare una libreria esterna che realizzasse il supporto a XMPP. Il fine di questa scelta è di avere una base solida e (sperabilmente) funzionante con la quale sviluppare il client. Compito del programmatore diventerebbe poi l'adattamento della libreria alla struttura del plugin, al fine di mantenere la compatibilità (vedi 3.1.3).

3.3 Libreria esterna: (a)Smack

La libreria più interessante tra quelle trovate al momento della decisione è stata Smack², della Ignite Realtime³, una compagnia di sviluppo di prodotti *open source* guidata da Jive Software⁴, una SBS⁵. Interessante e determinante nella scelta è stato sapere che Google ha utilizzato questa libreria per scrivere il supporto a Gtalk per il suo sistema operativo per *smartphone*: Google Android⁶, il cui logo è riportato in figura 3.2.



Figura 3.2: Il logo Android

Smack è una libreria opensource per sviluppare client XMPP. È scritta in java puro, come PariPari, ed è progettata per essere integrata in programmi più

²vedi [13]

³<http://www.igniterealtime.org/>

⁴<http://www.jivesoftware.com/about>

⁵Social Business Software

⁶<http://www.android.com/>

grandi, con l'obiettivo di fornire al programmatore un supporto completo a tutte le funzionalità di XMPP. La libreria è rilasciata sotto licenza Apache 2.0⁷, una licenza *free software* compatibile con la General Public License versione 3⁸.

3.3.1 Struttura di Smack

Smack ha il suo centro nella classe `XMPPConnection`, che si occupa, come dice il nome, di fornire una connessione a server XMPP. Tramite `XMPPConnection` è possibile effettuare la procedura di login, fornendo le credenziali del servizio tramite parametri al costruttore o metodi della classe. `XMPPConnection` fornisce l'accesso alla lista contatti tramite il metodo `XMPPConnection.getRoster()` che restituisce un oggetto di tipo `Roster`.

`Roster` rappresenta non solo una rappresentazione della lista contatti ma anche un vero e proprio collegamento della stessa: eventuali modifiche al `Roster` si ripercuotono su server grazie al lavoro di Smack. Questo dato è importante ai fini della codifica di IM in quanto le classi deputate alla gestione della lista contatti sono state originariamente pensate come meri contenitori. Grazie a questa classe, invece, le classi di `Xmpp` saranno contenitori "intelligenti", senza che si renda necessaria la scrittura di codice ulteriore per propagare le modifiche al server.

`Roster` fornisce l'accesso ai singoli contatti sotto forma di oggetti di tipo `RosterEntry`, anch'essa classe "intelligente" in quanto è possibile, ad esempio, richiedere lo stato aggiornato del contatto mediante una chiamata a metodo in un'istanza.

La classe `ChatManager`, della quale è possibile ottenere un'istanza mediante il metodo `XMPPConnection.getChatManager()`, è responsabile dell'istanziamento di nuove conversazioni. Una nuova conversazione, o chat, nel gergo Smack, è creabile attraverso il metodo `ChatManager.createChat()`, che restituisce un'istanza della classe `Chat`. Tale metodo accetta come parametri il JID del contatto da chiamare e un oggetto che implementa l'interfaccia `MessageListener`. Quest'ultima interfaccia descrive un cosiddetto *event listener* (o più concisamente *listener*), ascoltatore di eventi, infatti specifica un metodo, `processMessage()`, che verrà chiamato ogni qual volta un messaggio verrà ricevuto per quella chat. La classe

⁷<http://www.apache.org/licenses/LICENSE-2.0>

⁸vedi <http://gplv3.fsf.org/> e http://en.wikipedia.org/wiki/Apache_license#GPL_compatibility

3. IL PLUGIN IM, PIANIFICAZIONE DEL LAVORO E SVILUPPO DI XMPP

`Chat` consente di gestire una conversazione, in particolare il metodo sicuramente più interessante è quello per spedire messaggi alla controparte: `sendMessage()`.

Per intercettare particolari eventi provenienti dal server è possibile utilizzare oggetti di tipo `PacketListener` e `PacketFilter` da allacciare alla connessione mediante il metodo `XMPPConnection.addPacketListener()`. `PacketFilter` è un'interfaccia per filtri di pacchetti, di esso viene realizzato il metodo `accept()`, che restituisce un oggetto booleano vero se il pacchetto è del tipo desiderato, falso altrimenti. `Smack` ha al suo interno una serie di filtri già pronti, peraltro combinabili mediante classi che realizzano l' AND e OR logici tra filtri, per le configurazioni di filtraggio più comuni⁹. `PacketListener` è un'interfaccia per ascoltatori di eventi; viene definito il metodo `processPacket()` che viene chiamato ogni volta che il pacchetto transitante in `XMPPConnection` soddisfa il filtro associato all'istanza di `PacketListner`.

È possibile associare dei listener anche ad istanze di `Roster`, tali listener vengono evocati ad esempio in caso di aggiunta di nuovi contatti da parte di connessioni attive in client diversi, oppure al cambiamento di stato dei contatti. Grazie a questo genere di listener è possibile codificare comportamenti particolari dell'applicazione in reazione a eventi relativi alla lista contatti.

3.3.2 Problemi di Smack

`Smack` realizza quasi totalmente le funzionalità dichiarate nell'RFC di XMPP, tuttavia vi sono delle parti non complete o non funzionanti. Quelle di maggior interesse per IM sono l'autenticazione a mezzo SASL e, secondariamente, il supporto al meccanismo DNS/SRV¹⁰. Il mancato funzionamento del protocollo SASL impedisce, di fatto, la connessione a servizi come facebook e Gtalk, obiettivo del lavoro. Esiste la possibilità di aggirare il problema, almeno per quanto concerne Gtalk, attraverso dei gateway esterni. Questa soluzione però presenta delle problematiche di sicurezza non indifferenti, tra cui la necessità di divulgare le proprie credenziali a server esterni potenzialmente malevoli, inoltre IM, in que-

⁹vedi <http://www.igniterealtime.org/builds/smack/docs/latest/documentation/processing.html>

¹⁰vedi http://en.wikipedia.org/wiki/Domain_Name_System e http://en.wikipedia.org/wiki/SRV_record

sto modo, diventerebbe dipendente da questi server, rendendo inutile lo sforzo di decentralizzazione.

SRV è una funzionalità del servizio DNS che consente di sapere nome dell'host e numero di porta TCP al quale è erogato un certo servizio all'interno di un dominio. In particolare è utilizzato dai client XMPP per scoprire a che indirizzo e porta è raggiungibile il servizio XMPP relativo alla parte host di un JID (la parte dopo il simbolo "@"). Senza queste due funzionalità di fatto si perde la possibilità di raggiungere gli obiettivi preposti in questa tesi, tuttavia una soluzione c'è, ed è aSmack¹¹, una serie di *patch* patch scritte per Android volte a colmare le lacune di Smack.

3.3.3 Google Android e aSmack

aSmack è un progetto di patch alla libreria Smack per realizzare il supporto al meccanismo SASL e a DNS/SRV. Le patch sono state scritte con l'obiettivo di includere la libreria risultante all'interno del progetto Android. Android è un sistema operativo per smartphone sviluppato da Google, fortemente incentrato sulla fruizione di contenuti su Internet e sull'integrazione ai servizi di Google. In particolare Smack è stato utilizzato, assieme alla patch aSmack, nello sviluppo del client integrato per Gtalk.

3.4 Innesto in IM e interazione con altri plugin

Come descritto nel paragrafo 3.3.1 la struttura di Smack assomiglia molto all'architettura di infrastrutture definite in 3.1.3, per questo motivo il compito di integrare tale libreria in IM si è rivelato relativamente semplice.

Punto di partenza di Smack è la classe `XMPPConnection` e la prima questione è stata definire dove collocarla. Si è optato per includerla nella classe `XmppUser`, che realizza l'interfaccia `IUser`. Questa scelta è dovuta al fatto che è necessario attivare una connessione separata per ogni utente.

La classe `Roster` realizza quasi tutte le funzionalità descritte nell'interfaccia `IContacts`, si è quindi deciso di costruire una classe `XmppContacts` che contenesse al suo interno un'istanza di `Roster` prelevata da `XMPPConnection`. Tale scelta presenta vantaggi e svantaggi, in particolare le modifiche alla lista contatti, se

¹¹<http://code.google.com/p/asmack/>

3. *IL PLUGIN IM, PIANIFICAZIONE DEL LAVORO E SVILUPPO DI XMPP*

opportunamente mappate sulle corrispondenti funzioni di `Roster`, vengono propagate al server senza problemi. Il fatto che la classe sia così legata all'oggetto `Roster` ne complica però la gestione, ad esempio: nel caso venga richiesto l'elenco dei soli utenti online, o qualunque sottoinsieme dei contatti, è necessario fornire un nuovo oggetto, sempre realizzante l'interfaccia `IContacts`, che contenga quei contatti. Questo è una diretta conseguenza del fatto che `Roster` propaga le modifiche al server.

Si è scelto di scrivere la classe `XmppContact`, che realizza l'interfaccia `IContact`, all'interno della classe `XmppContacts`, al fine di facilitare la scrittura di quest'ultima, specie nel caso di interazione con funzioni proprie di `XmppContacts`.

Gli oggetti `Chat` possono essere memorizzati in strutture dati all'interno di `XmppUser`, in quanto appartengono al singolo utente. Ad ogni `Chat` deve essere fornito un listener per i messaggi in arrivo, nello specifico tale listener deve provvedere ad istanziare una finestra di conversazione e deve scrivere il messaggio arrivato a video. Per realizzare ciò si è pensato di elaborare una classe a sè stante: `XmppChatMessageListener`.

La classe `XmppClient` si occuperà solo di istanziare nuovi utenti e toglierli dalla lista una volta che è stato effettuato il logout. Non sarà necessario realizzare il codice relativo al passaggio di messaggi attraverso la classe `XmppClient` perchè tale parte è nascosta dalla libreria¹².

Definiti questi “punti di contatto” tra le classi di Smack e di IM è possibile passare alla realizzazione.

¹²in realtà andrebbe realizzato per conformità alla struttura di PariPari, maggiori dettagli al paragrafo 4.5.1

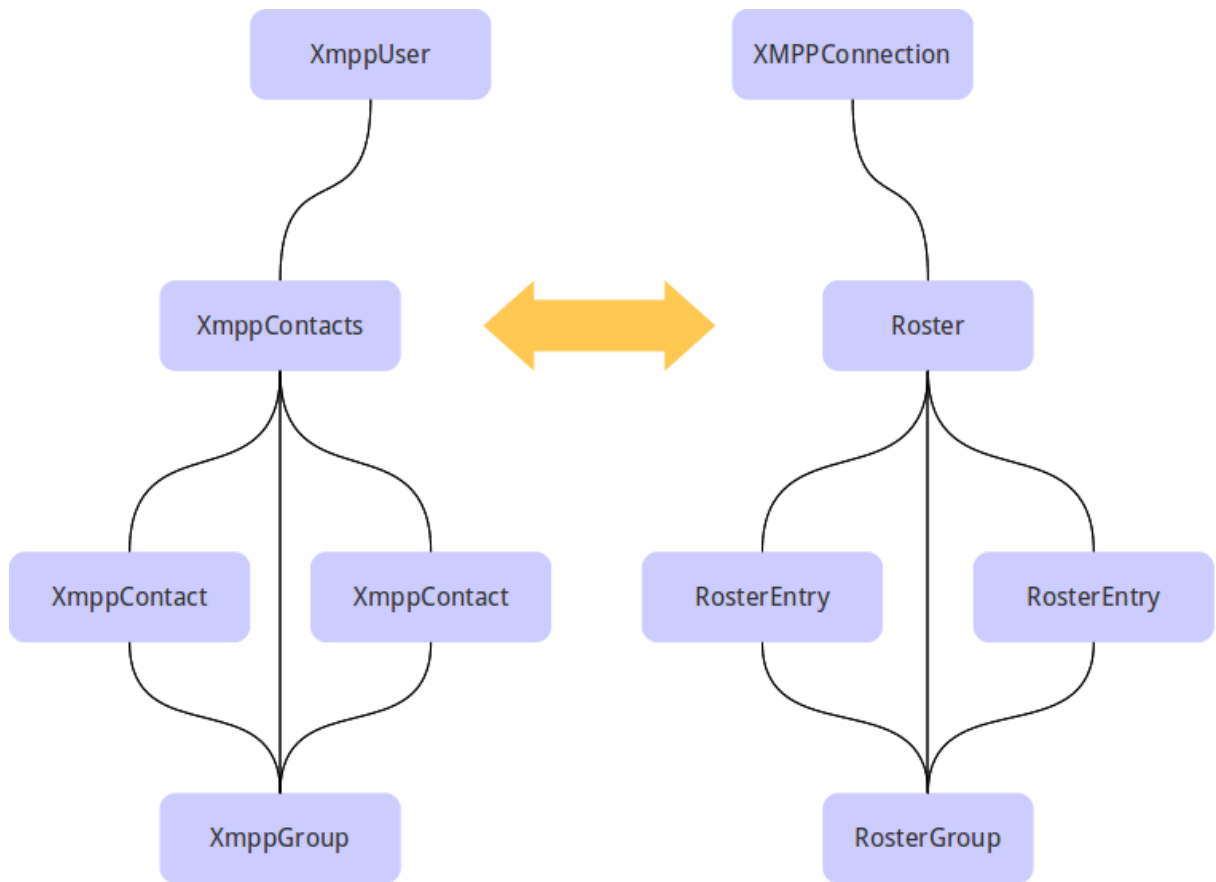


Figura 3.3: Analogie di architettura tra IM e Smack

3. *IL PLUGIN IM, PIANIFICAZIONE DEL LAVORO E SVILUPPO DI XMPP*

Capitolo 4

Realizzazione

Con riferimento alla sezione 3.4 vediamo come sono state implementate le classi principali del supporto a XMPP.

4.1 XmppUser

Analizziamo le parti salienti di `XmppUser` a cominciare dal metodo `login()`: nel listato 4.1 vediamo un estratto di tale metodo. Nell'estratto si vede, alle righe 139 e 141 come sia possibile configurare la connessione al fine di connettersi a nomi host particolari e porte non standard.

Altro pezzo di codice saliente è quello relativo ai messaggi in arrivo, sempre all'interno del metodo `login()`: il listato 4.2 ne mostra l'estratto. Alla riga 168 viene istanziato un oggetto `PacketFilter` per tipo il di pacchetto `Message`. Tale oggetto risponde con un booleano "true" al metodo `accept()` nel caso nella connessione passi, appunto, un messaggio diretto all'utente. Alla riga 171 viene realizzato il metodo `processPacket()` della classe anonima `PacketListener`, tale metodo viene invocato ad ogni messaggio ricevuto nella connessione. Questo metodo si occupa di verificare che sia istanziato un oggetto di tipo `Chat` ogni volta che arriva un messaggio da un certo contatto, sempre che l'oggetto non sia già stato istanziato per quella conversazione. Nel caso l'oggetto `Chat` per la conversazione non esista provvede a crearlo lanciando il metodo `createChat()` che si occupa, tra l'altro, di istanziare il listener per i messaggi di quella sessione di *im*, e successivamente si invoca il listener della conversazione al fine di mostrare il primo messaggio arrivato. In questo modo se l'oggetto `Chat` è stato già

4. REALIZZAZIONE

```
130 public void login() throws NotConnectedException{
131     try{
132         if(conn==null || !conn.isConnected()){
133
134             if(host.equals("talk.google.com")){
135                 //conf=new ConnectionConfiguration(host,port,"gmail.com");
136                 //conf=new ConnectionConfiguration("gmail.com",5222);
137                 conf=new ConnectionConfiguration("gmail.com");
138             }else{
139                 conf=new ConnectionConfiguration(host,port);
140             }
141             conn=new XMPPConnection(conf);
142             conn.connect();
143             System.out.println("Connected");
144         }
145
146         if(!conn.isAuthenticated()){
147             //connect() launches login, but it can fail
148             //conn.login(username,password,resource);
149             if(host.equals("talk.google.com")){
150                 conn.login(getEmail(),password,resource);
151             }else{
152                 conn.login(username,password,resource);
153             }
154         }
155         conn.sendPacket(new Presence(Presence.Type.available));
156     }catch(XMPPException e){
157         throw new NotConnectedException();
158     }
```

Listato 4.1: XmppUser: estratto del metodo login()

```
168 //instantiate a generic message listener
PacketFilter filter=new PacketTypeFilter(Message.class);

170 PacketListener myListener=new PacketListener(){
    public void processPacket(Packet packet) {
172         StringTokenizer tok=new StringTokenizer(packet.getFrom(), "/");
        String from=tok.nextToken();
174         tok=null;
        tok=new StringTokenizer(packet.getTo(), "/");
176         String to=tok.nextToken();
        tok=null;
178         if(to.equals(mail)){
            Chat c=chats.get(from);
180             if(c==null){
                //the chat doesn't exist, so create it
182                 createChat(from);

184                 //then run the chat message listener
                c=chats.get(from);
186                 c.getListeners().iterator().next()
                    .processMessage(c,(Message) packet);
188             }
            //if here then the chat already exists,
190             // so the message is already parsed

192         }else{
            //if here there's an error, a connection can
194             //only receive messages that are directed to the user.
            System.out.println("Receiving a different message: from: "+
196                 from+ " to: "+ to+ " instead of: "+mail);
        }
198     }
200 };

202 conn.addPacketListener(myListener, filter);
```

Listato 4.2: XmppUser: il filtro per i messaggi in arrivo

4. REALIZZAZIONE

istanziato per quel contatto non è necessario fare alcunchè, infatti provvederà il listener fornito a mostrare i messaggi. Alla riga 202 il listener di messaggi, viene “agganciato” alla connessione `XMPPConnection`, unitamente al filtro di cui prima.

Vediamo ora l’estratto relativo alla spedizione di messaggi: il listato 4.3 mostra il metodo `sendMessage()`. Tale metodo si occupa di spedire messaggi ad un

```
244  /**
245   * Send a message to a contact
246   * @param message the message to send
247   * @param contact the contact to send the message to
248   * @throws OperationNotPerformedException on error while sending message;
249   */
250  public void sendMessage(String message,String contact) throws
      OperationNotPerformedException{
251      Chat c=chats.get(contact);
252      if(c==null){
253          createChat(contact);
254          c=chats.get(contact);
255      }
256      try{
257          c.sendMessage(message);
258      }catch(Exception e){
259          System.out.println("Error sending message...");
260          //handle this please ;)
261          throw new OperationNotPerformedException("Cannot send message");
262      }
263  }
```

Listato 4.3: XmppUser: il metodo `sendMessage()`

particolare contatto. Se tale per tale contatto non esiste una chat essa viene creata (vedi riga 253) prima di spedire il messaggio.

Alcuni metodi non sono stati scritti, in particolare il metodo `addNewGroup()` è impossibile da scrivere in quanto un gruppo non è un’entità definibile in XMPP. Un gruppo in quanto tale non esiste, però è possibile aggiungere ad ogni contatto una serie di etichette di appartenenza a gruppi. In questo senso un operazione concettualmente semplice come la ri denominazione di un gruppo comporterebbe, nella pratica, a dover ricercare tutti i contatti appartenenti al vecchio gruppo, togliere loro l’attributo di appartenenza al vecchio gruppo e aggiungere quello del nuovo. Fortunatamente tali macchinose procedure sono effettuate dalla classe `RosterGroup` di Smack, che tuttavia funziona solo per gruppi preesistenti, ovvero solo dopo che si è aggiunto l’attributo di appartenenza ad almeno un contatto (vedi 4.3 per maggiori dettagli).

Un altro genere di funzioni non scritte in `XmppUser` sono quelle relative al recupero di sottoinsiemi della lista contatti, ad esempio `getOfflineList()` e `getOnlineList()`. Tali funzioni sono realizzate dalla classe `XmppContacts`, e su `XmppUser` viene chiamato il metodo di quest'ultima classe. La motivazione di tale scelta è nella struttura di `XmppContacts`, fortemente legata alla classe `Roster`, come spiegato nella sezione 3.4 e nella 4.2.

4.2 XmppContact / XmppContacts

La classe `XmppContacts` realizza l'interfaccia `IContacts`, e al suo interno viene implementata la classe `XmppContact` che realizza invece l'interfaccia `IContact`.

La classe `Roster` mette a disposizione la possibilità di assegnare un listener per monitorare i cambiamenti dell'elenco contatti e i cambiamenti di stato. In `XmppContacts` tale listener è stato pensato in maniera tale che ad ogni evento della lista contatti esso invochi l'aggiornamento dell'oggetto `Roster`, ovvero far sì che venga scaricata di nuovo l'intera lista contatti. Tale soluzione non solo è banale ma, osservando i dati dei test preliminari, provoca un eccessivo numero di scaricamenti della lista. Per ovviare a ciò è stato posto un limite al numero di aggiornamenti per unità di tempo. La soluzione è descritta nel listato 4.4.

```
496     private void reloadRoster(){
498         //try to deal with too frequent reloading.
500         long curr=System.currentTimeMillis();
502         if(last==null){
504             roster.reload();
506             last=new Date(curr);
508         }else if (last.getTime()<curr-60000){
510             roster.reload();
512             last.setTime(curr);
514         }
516     }
```

Listato 4.4: `XmppContacts`: il metodo `reloadRoster()` interno al listener dei cambiamenti della lista contatti

Per ovviare alle necessità di `XmppUser` di avere un sottoinsieme della lista contatti con determinate caratteristiche è stata scritta una classe interna privata.

Tale classe, chiamata `XmppContactsContainer`, è stata costruita per essere un mero contenitore, come previsto dalle interfacce. Per scongiurare interazioni non consentite o prive di significato, per esempio il tentativo di aggiungere un contatto alla lista di contatti online, è stato implementato un meccanismo di blocco della classe. La motivazione di ciò è il fatto che l'elemento `Roster`, e quindi l'intera classe `XmppContacts`, è sincronizzato con l'elenco contatti su server. Una volta inseriti i contatti del sottoinsieme è possibile impedire altri inserimenti mediante una funzione detta `lock()`.

Tramite questa classe sono stati implementati i metodi relativi a sottoinsiemi di utenti all'interno di `XmppContacts` anzichè in `XmppUser`. Un esempio sono i metodi `getOnlineList()` e `getOfflineList()`, il cui codice è riportato nel listato 4.5.

```
672 public IContacts getOnlineList(){
    XmppContactsContainer ol=new XmppContactsContainer();
674     for(XmppContact c:(XmppContact [])toArray()){
        if(c.isOnline())
            ol.addContact(c);
676     }
    ol.lock();
678     return ol;
    }
680
682 public IContacts getOfflineList() {
    XmppContactsContainer ol=new XmppContactsContainer();
684     for(XmppContact c:(XmppContact [])toArray()){
        if(!c.isOnline())
            ol.addContact(c);
686     }
    ol.lock();
688     return ol;
    }
```

Listato 4.5: `XmppContacts`: i metodi `getOnlineList()` e `getOfflineList()`

4.3 `XmppGroup`

La particolarità di `XmppGroup` è che la classe non è istanziabile se non esiste almeno un utente che appartenga a quel gruppo. Questo è dovuto, come nelle altre classi, all'architettura di XMPP e a quella delle classi di Smack.

`XmppGroup` è fortemente legato alla classe `RosterGroup`, la classe di Smack per gestire gruppi di contatti. Aggiungere un contatto a un `RosterGroup` equivale ad assegnare un attributo al contatto, non esiste cioè una “entità” gruppo, ma solo una collezione di contatti con lo stesso valore nell’attributo. Questa caratteristica è così particolare che operazioni come la ri-denominazione del gruppo rendono di fatto irraggiungibile l’oggetto `XmppGroup`. Questo è ovviabile fornendo a `XmppGroup`, in fase di costruzione, un riferimento all’oggetto `Roster`, tramite questo infatti è possibile fare in sequenza i seguenti passi:

- Rinominare il gruppo tramite i metodi di `RosterGroup`
- Acquisire, dall’oggetto `Roster`, l’oggetto `RosterGroup` corrispondente al nuovo nome del gruppo
- Ri-assegnare alla variabile interna `RosterGroup` il riferimento all’attuale nome del gruppo.

A questo punto l’oggetto ha come contatti sia i vecchi contatti sia, eventualmente, quelli che già appartenevano al gruppo del nuovo nome.

4.4 Difficoltà incontrate

La comodità di avere a disposizione una libreria già pronta comporta vantaggi e svantaggi. Lo svantaggio principe è quello di non avere la certezza che, specie in progetti ampi, vengano coperte tutte le funzionalità pubblicizzate. Purtroppo Smack ha qualche difetto, nella fattispecie al momento di testare la libreria, ad uno stadio di sviluppo ormai avanzato, è emerso che defice del supporto completo a SASL. Questa anomalia, purtroppo, oltre ad essere stata scoperta tardi è anche bloccante per lo sviluppo del plugin in quanto SASL è obbligatorio per la connessione ai servizi XMPP di Gtalk e della chat di facebook. In nell’RFC di XMPP è specificato che un server può supportare diversi meccanismi di autenticazione SASL, tuttavia Smack non supporta proprio quelli utilizzati da Gtalk e facebook. Questo problema ha portato a non completare, di fatto, lo sviluppo del supporto completo a XMPP, tuttavia, almeno per i server che non utilizzano SASL, il *backend* di supporto è funzionante. L’insieme di patch di aSmack dovrebbe servire ad ovviare a questi problemi, tuttavia nonostante i tentativi, non si è riusciti a

integrare la nuova libreria, sia per il poco tempo a disposizione sia per la scarsa documentazione disponibile.

A rallentare ulteriormente lo sviluppo ha contribuito la difficoltà di interazione con la GUI¹ di IM. Tale GUI è solo un'interfaccia temporanea in attesa del plugin GUI di PariPari, questo e la mancanza di documentazione a riguardo sono stati i motivi per cui ad oggi non è possibile fruire del completo supporto a XMPP per via grafica.

4.5 Sviluppi futuri

Per quanto il lavoro finora svolto sia funzionante, a meno delle limitazioni sopra descritte, è possibile delineare alcune strade di miglioramento ed estensione delle funzionalità del plugin.

4.5.1 Uso delle strutture socket del plugin Connectivity

In PariPari il Core impedisce, per ragioni di sicurezza, l'uso di uno specifico sottoinsieme di classi Java, tra cui la classe `Socket`. Questo pone dei problemi nell'uso di classi esterne, come Smack, ne fanno uso. Fortunatamente Smack offre la possibilità di fornire alla classe `XMPPConnection` un oggetto di tipo `SocketFactory`², in questo modo è possibile iniettare i socket di PariPari nella libreria in maniera trasparente.

4.5.2 Uso del sistema di messaggi interno di IM

IM, al suo interno utilizza uno schema di invio e ricezione messaggi basato su linee successive di testo. Tale sistema, sebbene non particolarmente adatto allo *streaming* di pacchetti basati su XML, è utilizzabile e funzionante e consente a `ImConnectionManager` di gestire in maniera centralizzata tutte le connessioni attive. Per integrare Smack in questo sistema è possibile inserire, oltre alla `SocketFactory`, un'ulteriore livello intermedio. Dato che verosimilmente le classi di Smack utilizzeranno gli *stream* in ingresso ed in uscita dai socket è possibile

¹Graphic User Interface

²`javax.net.SocketFactory`, vedi Javadoc corrispondente su <http://download.oracle.com/javase/6/docs/api/index.html>

simulare tali oggetti all'interno di false classi `Socket`. Questi oggetti *stream* personalizzati convertiranno il flusso XML in flussi basati su righe di testo, per poi inviarli ai listener di IM tramite `ImConnectionManager`.

4.5.3 Trasferimento file

XMPP prevede delle estensioni per il trasferimento di file. Tale funzionalità è fortemente desiderata in quanto amplierebbe di molto le funzionalità offerte dal client. Smack offre un parziale supporto a tale funzionalità ed è desiderabile che venga implementata in un futuro prossimo.

Un altro approccio sarebbe quello di affidarsi al plugin `DistributedStorage`, tuttavia questo richiede che entrambe le parti dello scambio siano sulla rete di PariPari.

4.5.4 Sfruttamento della funzionalità gateway di XMPP e sviluppo di un server dedicato

Come descritto nella sezione 2.2.6 XMPP offre la possibilità di sfruttare server esterni per fornire il supporto a protocolli di comunicazione di terze parti. Per ampliare il parco protocolli di IM è possibile valutare l'ipotesi di utilizzare, per lo meno temporaneamente, questo tipo di servizio.

In seconda battuta si può pensare di sviluppare un server XMPP distribuito sulla rete paripari. Tale progetto potrebbe includere i gateway per i vari protocolli, offrendo la possibilità di scrivere il plugin IM ottimizzandolo per XMPP e lasciando al server distribuito il compito di tradurre i messaggi per le reti di terze parti.

4.5.5 Un protocollo PariPariIM

Da tempo è stata ventilata l'eventualità di studiare e implementare un nuovo protocollo di *im*. L'architettura di tale protocollo dovrebbe essere progettata per adattarsi bene all'architettura di PariPari. In particolare, vista la natura p2p della piattaforma, è auspicabile che non necessiti di server (seppur distribuiti) per funzionare. In linea con la filosofia di PariPari, poi si potrebbe implementare un meccanismo di integrazione tra i protocolli esistenti. Come illustrato nella figura 4.1 il plugin IM dovrebbe porsi come intermediario in eventuali conversa-

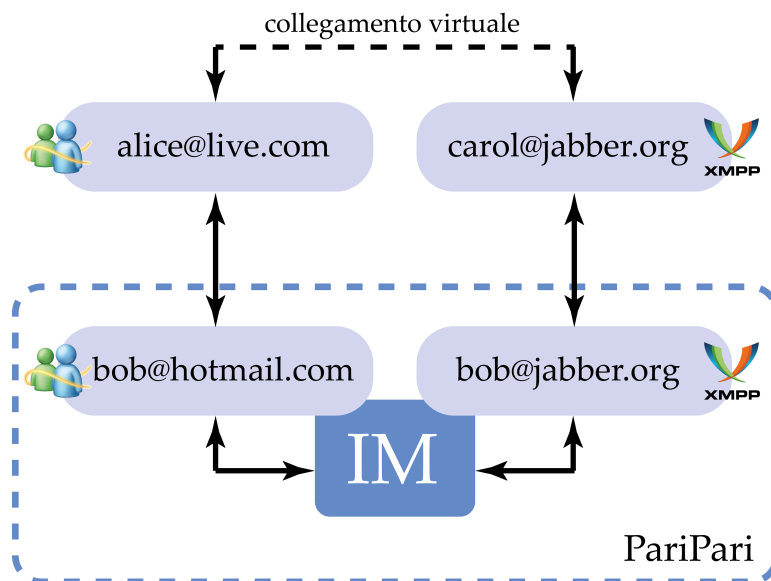


Figura 4.1: Uso di IM come integrazione tra protocolli

zioni inter-protocollo. Nell'esempio Bob è in conversazione con Alice e Carol, il plugin IM di Bob permette una conversazione a protocolli eterogenei altrimenti impossibile. Una via per realizzare questo meccanismo potrebbe essere quella di realizzare un sistema di meta-contatti. Un meta-contatto è un'astrazione di un conoscente: di una persona si possiedono diversi contatti, e in genere si è interessati a contattare tale persona, piuttosto che preoccuparsi della rete alla quale è connessa. Il plugin IM dovrebbe preoccuparsi di valutare, di volta in volta, su quale rete il conoscente è disponibile e dirottare la conversazione su tale rete, in maniera totalmente trasparente all'utente.

Capitolo 5

Conclusioni

In questo documento è stata vista la progettazione e realizzazione del supporto al protocollo XMPP per il plugin IM del progetto PariPari. Con in mente i principi e gli obiettivi di PariPari si è visto che il protocollo XMPP è la scelta migliore. In primo luogo, essendo alla base dei servizi di messaggistica più diffusi attualmente, permette di attrarre potenzialmente un elevatissimo numero di utenti sin da principio. A suffragio di tale scelta ci sono poi molti fatti concreti, ad esempio il fatto che il protocollo sia uno standard aperto e approvato dalla IETF. Inoltre XMPP garantisce nativamente funzionalità non scontate come il login da postazioni multiple, la cifratura del flusso di messaggi e l'autenticazione sicura. Secondariamente, tramite il servizio *gateway* consente, a patto di appoggiarsi a server esterni, la connessione a reti di terze parti senza necessità di adattamenti o ristrutturazioni del codice. Tenendo conto dell'impostazione ad alto livello del plugin IM, e della presenza in rete di librerie di supporto a XMPP con una struttura in buona parte sovrapponibile all'architettura interna, il lavoro di sviluppo è stato velocizzato, non dovendosi preoccupare della mera traduzione delle specifiche e di altri dettagli realizzativi. D'altra parte la scarsità di documentazione a riguardo ha comportato delle difficoltà nel correggere le mancanze della libreria utilizzata.

Nonostante le limitazioni, crediamo che la realizzazione del supporto a XMPP possa essere punto cardine per la penetrazione pervasiva di PariPari nel mercato. Con questo lavoro ci auguriamo di aver fornito le basi per ampliare il ventaglio di servizi offerto da PariPari, con la speranza che questo possa contribuire al successo del progetto.

5. *CONCLUSIONI*

“Io ho quel che ho donato” (G. D’annunzio)

BIBLIOGRAFIA

Bibliografia

- [1] Paolo Bertasi, *Progettazione e realizzazione in java di una rete peer to peer anonima e multifunzionale*, Dipartimento d'Ingegneria dell'Informazione, Università di Padova, 2004
- [2] Giorgio Vallini, *Progettazione e realizzazione in java di un client instant messaging multi-protocollo*, Dipartimento d'Ingegneria dell'Informazione, Università di Padova, 2009
- [3] Wikipedia contributors, “World Wide Web”, *Wikipedia, The Free Encyclopedia*, http://en.wikipedia.org/w/index.php?title=World_Wide_Web&oldid=394765069 (in data 5 novembre 2010).
- [4] Wikipedia contributors, “Internet Relay Chat”, *Wikipedia, The Free Encyclopedia*, http://en.wikipedia.org/w/index.php?title=Internet_Relay_Chat&oldid=394966947 (in data 5 novembre 2010).
- [5] Wikipedia contributors, “Instant messaging”, *Wikipedia, The Free Encyclopedia*, http://en.wikipedia.org/w/index.php?title=Instant_messaging&oldid=394789625 (in data 5 novembre 2010).
- [6] Wikipedia contributors, “Facebook features”, *Wikipedia, The Free Encyclopedia*, http://en.wikipedia.org/w/index.php?title=Facebook_features&oldid=395046494#Chat (in data 5 novembre 2010).
- [7] Wikipedia contributors, “Google Talk”, *Wikipedia, The Free Encyclopedia*, http://en.wikipedia.org/w/index.php?title=Google_Talk&oldid=394563592 (in data 5 novembre 2010)
- [8] Wikipedia contributors, “Extensible Messaging and Presence Protocol”, *Wikipedia, The Free Encyclopedia*, <http://en.wikipedia.org/w/index>.

BIBLIOGRAFIA

- [php?title=Extensible_Messaging_and_Presence_Protocol&oldid=391014848](#) (in data 5 novembre 2010).
- [9] XMPP Standard foundation, *The XMPP Standard foundation*, www.xmpp.org (in data 17 novembre 2010)
- [10] Wikipedia contributors, “Windows Live Messenger”, *Wikipedia, The Free Encyclopedia*, http://en.wikipedia.org/w/index.php?title=Windows_Live_Messenger&oldid=394704951 (in data 5 novembre 2010)
- [11] Wikipedia contributors, “Kademlia”, *Wikipedia, The Free Encyclopedia*, <http://en.wikipedia.org/w/index.php?title=Kademlia&oldid=393934831> (in data 8 novembre 2010)
- [12] World Wide Web Consortium (W3C) , “Extensible Markup Language (XML)”, <http://www.w3.org/XML/> (in data 17 novembre 2010)
- [13] Ignite Realtime, “Smack API”, <http://www.igniterealtime.org/projects/smack/index.jsp>, (in data 18 novembre 2010)