

Apple Rendezvous e Zeroconf

Luca Caretoni
IkkiSoft - www.ikkisoft.com

7 ottobre 2007



Questo documento vuole fornire una descrizione sull'architettura della tecnologia OpenSource Zeroconf commercializzata da Apple¹ con il nome Rendezvous (ora *Bonjour*), cercando di analizzare come l'utilizzo di questa tecnologia influisce in maniera positiva per l'utente nell'interfacciamento veloce di dispositivi eterogenei e considerando inoltre le potenzialità messe a disposizione degli sviluppatori.

¹Apple e tutti i prodotti citati nel presente testo sono da riferirsi ai legittimi proprietari. Gran parte delle informazioni contenute deriva infatti dal sito web <http://developer.apple.com/macosx/rendezvous/>.

Copyright (c) 2004 Luca Carettoni - IkkiSoft

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.



1 Introduzione

Nel maggio 2002 Apple annunciò la presenza del protocollo di rete Open-Source *Zeroconf* all'interno del nuovo sistema operativo Mac OS X 10.2 (Jaguar), commercializzando questa tecnologia con il nome di **Rendezvous**.

Rendezvous permette il discovery di computers, devices e servizi su reti Ip. L'utilizzo di tale standard come supporto alla comunicazione è giustificato dalla sua ormai consolidata presenza nel campo delle reti di elaboratori. Attraverso le estensioni introdotte da Rendezvous è possibile effettuare la ricerca di dispositivi in rete senza dover conoscere ed immettere l'indirizzo ip o dover configurare Dns server.

Rendezvous lavora quindi con tutti i maggiori standard di connessione attuali incluso Ethernet, Airport e Airport Extreme (nome commerciale di Apple per la tecnologia 802.11b/g) e grazie al supporto fornito da numerose aziende tra cui Brother, Canon, Hp, LexMark, Xerox oltre a Philips e Aspyr potrebbe diventare di fatto uno standard per l'interoperabilità "*zero configuration*" di sistemi eterogenei.

Rendezvous è un protocollo aperto, ed è stato approvato dalla IETF (Internet Engineering Task Force) che si occupa di definire standard in fatto di interoperabilità.

Per fornire tutte le potenzialità del protocollo, Rendezvous richiede che i software o i devices che implementano la tecnologia debbano fornire tre componenti essenziali, che permettono al dispositivo di:

- acquisire in maniera automatica un indirizzo ip senza l'uso di Dhcp server
- tradurre senza l'uso di Dns server, gli indirizzi ip dei dispositivi in nomi identificativi
- ricercare o fornire servizi senza l'uso di un direttorio principale

Nei prossimi paragrafi approfondiremo la tecnologia sia dal punto di vista dell'utente finale, ma anche dal punto di vista dello sviluppatore al quale Rendezvous offre nuove opportunità nello sviluppo di applicazioni.

2 Rendezvous per l'utente

Rendezvous è pensato in primo luogo per gli utenti in maniera da semplificare notevolmente l'installazione e l'utilizzo di dispositivi e servizi diversi. Il supporto alle reti wifi permette di ampliare notevolmente le prospettive della tecnologia aprendo le strade verso una condivisione di servizi dinamici in maniera flessibile.



Figura 1: Interoperabilità tra sistemi eterogenei.

Ipotizziamo ora un possibile scenario di utilizzo mettendo in luce gli aspetti chiave di Rendezvous.

Una normale giornata di lavoro: Steve entra in ufficio, estrae dalla borsa il portatile ed è subito connesso ad internet. Rendezvous si è occupato di assegnargli un ip (senza l'uso di DhcP server) e di collegarlo alla rete wireless dell'azienda. A questo punto Steve deve finire di scrivere un articolo insieme ad un suo collaboratore che ha l'ufficio al piano di sopra dell'edificio. Apre SubEthaEdit,² un editor di testo studiato appositamente per l'interazione tramite Rendezvous, ed inizia a lavorare in remoto con il suo collega sull'articolo. Nessuno dei due ha dovuto configurare niente, eseguendo il software Rendezvous si è accorto di tutti coloro che all'interno della rete utilizzano quel particolare servizio e ne ha permesso l'associazione in maniera automatica. Tra una canzone e l'altra (eseguite da remoto tramite un player Mp3

²SubEthaEdit è un ottimo strumento che ben rappresenta la categoria di software sviluppati per Rendezvous. Per maggiori informazioni riferirsi all'indirizzo web <http://www.codingmonkeys.de/subethaedit>

che supporta la tecnologia) il lavoro è in breve terminato. Steve apre il pannello di stampa, gli si presentano tutte le stampanti Rendezvous compatibili e a questo punto può stampare senza problemi.

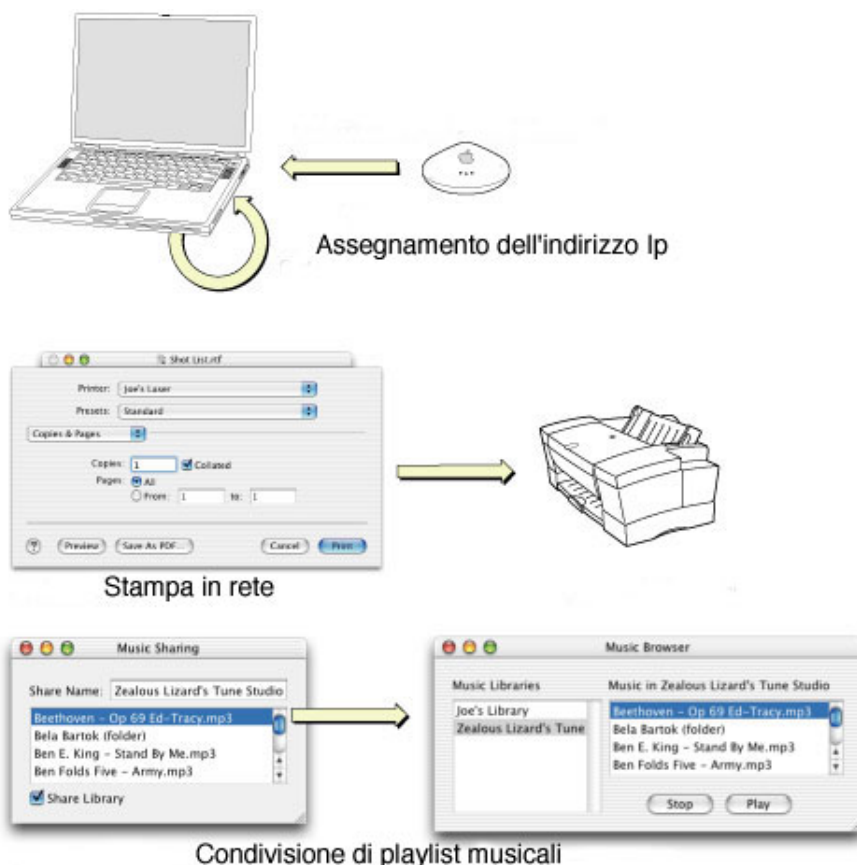


Figura 2: Alcune operazioni svolte nello scenario precedente.

Il discovery poteva essere effettuato nello stesso modo per qualsiasi tipo di servizio e di modalità di fruizione: dallo sharing di musica o di immagini, all'uso di applicazioni per l'instant messaging, alla condivisioni di dischi remoti oltre al discovery automatico da parte del browser di tutti i web servers all'interno della propria sottorete.

Effettuando un'astrazione di questo tipo sicuramente Rendezvous risulta una tecnologia funzionale e sempre più irrinunciabile; in ambiti come questo è però sempre necessario considerare la disponibilità di dispositivi e di software che effettivamente implementano questa tecnologia. Dal lato

suo, Apple fornisce già un'innumerabile quantità di software ma anche di hardware perfettamente compatibile e inoltre la scelta di utilizzare un protocollo OpenSource assicura l'interoperabilità tra sistemi eterogenei. Esistono già applicativi sviluppati per piattaforme Unix-like (Linux in particolare) e Windows che possono integrarsi completamente in questo contesto e con la presenza di aziende del settore interessate, la situazione non può che migliorare.

3 Architettura

Rendezvous è per prima cosa un protocollo, e come tale definisce delle linee guide per risolvere dei problemi in questo caso legati alla connessione e alla condivisione di servizi in rete. I principali aspetti da considerare sono:

Addressing: Assegnare indirizzi ip ad ogni host, in maniera automatica

Naming: Utilizzare come identificativi nomi invece che indirizzi ip

Discovery: Trovare e usare servizi sulla rete

Rendezvous risolve il problema legato all'Addressing attraverso l'uso del "Self-Assigned Ip" sullo stesso link. Prima di introdurre il funzionamento di tale meccanismo è necessario definire con precisione il significato di *link*. Due hosts sono sullo stesso *link* se nell'invio di pacchetti da un host all'altro, l'intero link payload non risulta variato ovvero in pratica se in una rete ethernet non esistono router che modificano il pacchetto durante il tragitto tra le due macchine. Ogni host di fatto possiede una visione locale della rete denominata *link-local range* e all'interno di questo "spazio di rete" si assegna in maniera autonoma un indirizzo Ip. Nel caso esista all'interno dello stesso link-local range un conflitto, l'ultimo host che si è assegnato l'indirizzo lo modifica. Un indirizzo di tipo link-local è un ip nel range 169.254/16 (esempio: 169.254.30.23) Questa modalità di assegnamento non è niente di innovativo se si pensa che risulta già presente nella maggior parte dei sistemi operativi da Mac OS 8.5 e Windows 98. Entrambi i protocolli IPv4 e IPv6 permettono l'assegnamento automatico degli indirizzi durante la fase di startup.

Consideriamo ora il problema legato al Naming. La proposta di Rendezvous è l'utilizzo del multicast DNS (mDNS) che si realizza con l'invio di pacchetti dns attraverso il protocollo ip multicast, evitando così l'uso di un server centrale di gestione. Ogni host, all'interno del rete si assegna un proprio *local hostname* definito durante la fase di configurazione e condivide i propri servizi identificato da quel nominativo. Non esiste in questo caso, a differenza del normale concetto di DNS hostname, un'autorità globale che amministra la gestione dei nomi, bensì ogni host fa per se. Conseguenza di

ciò è che il nome assegnato è valido solamente all'interno della propria sottorete locale e non ha rilevanza all'esterno da essa. Per differenziare questo concetto di visibilità rispetto al normale utilizzo degli hostname, si aggiunge il suffisso *.local* in seguito al nome dell'host (Esempio: *myHost.local*)

Quando noi digitiamo dal browser un normale indirizzo web la ricerca dell'host associato a quel particolare url è realizzata attraverso l'interrogazione a un dns server in qualche parte del mondo. Digitando invece *hostname.local* viene eseguita sul multicast DNS locale: Il client invia sulla rete la richiesta e riceve le risposte da tutti quegli hosts che hanno in esecuzione un mDNS responder.

L'ultimo aspetto riguarda la gestione del Discovery che permette alle applicazioni di ricercare i servizi disponibili, definendo i riferimenti per l'utilizzo di tale servizio (locazione, numeri di porte, ecc). Rendezvous usa anche in questo caso il multicast DNS per archiviare informazioni all'interno di record.

Rendezvous adotta un approccio nuovo all'uso e alla gestione dei dispositivi in rete e sposta l'approccio classico del considerare una rete come insieme di dispositivi in un **raggruppamento di servizi**. Per Rendezvous tutti i dispositivi sia hardware che software sono considerati come fornitori di servizi e questo aiuta notevolmente la visione dell'utente che non dovrà più preoccuparsi di quali macchine ci sono nella rete, ma semplicemente se esistono servizi che svolgono il lavoro per cui è interessato.

Esistono infatti esempi che rendono chiaro il modo di vedere introdotto da questo nuovo approccio.

Pensiamo ad un sito web, noi siamo interessati a poter fruire dei contenuti e poco ci importa se (come è probabile per siti di grandi dimensioni) tutto il sito è replicato da differenti webserver. Il sito web è per noi un servizio. Sempre rispetto all'esempio precedente possiamo intuire come risulti comodo per l'amministratore di rete spostare il servizio su dispositivi diversi senza introdurre blocchi del servizio stesso.

Prima di entrare in dettaglio in ognuno degli aspetti precedenti, è opportuno ricordare che Rendezvous introduce inoltre dei meccanismi di riduzione del traffico per limitare l'overhead introdotto in rete dall'uso dell' mDNS. Esistono funzionalità di *caching* per la ricerca dei servizi che evita ad ogni

hosts di richiedere l'elenco di un particolare servizio, ma una volta inviata la richiesta da parte di uno di questi, l'informazione rimane memorizzata anche per gli altri. Le richieste di informazioni legate ai servizi (service query) sono gestite in maniera da eliminare i duplicati e da ridurre i tempi di invio delle richieste in maniera esponenziale evitando così di sovraccaricare la rete. Del resto però l'introduzione di un nuovo host che offre un servizio precede l'invio di un messaggio di *ServiceAnnouncement* che velocizza l'uso per eventuali host che aspettavano proprio quel particolare servizio.

3.1 Rendezvous Domain

Abbiamo visto precedentemente che Rendezvous fornisce funzionalità a livello di *link-local range* ovvero tra host il cui scambio di pacchetti di rete non implica la modifica dell'header del pacchetto stesso. Il *.local* però non rappresenta realmente un dominio poichè si differenzia dai dns convenzionali per quanto riguarda l'univocità del nome. Ogni dns name del mondo è unico mentre anche all'interno dello stesso edificio possono esistere due host con nome uguale a patto che tali host non facciano parte dello stesso local-link. I suffissi locali non definiscono quindi limitazioni se non quella di evitare conflitti all'interno dello stesso local-link attraverso la ridenominazione automatica o manuale (la prima nel caso di dispositivi hardware senza schermo) del nome dell'host immesso nella rete più di recente.

Per la denominazione dei servizi già esistenti Rendezvous si rifà alle convenzioni ormai associate dell'Internet Assigned Numbers Authority (IANA) la quale mantiene traccia di tutti i protocolli registrati, con le relative porte utilizzate e le relative caratteristiche.

Un servizio sarà quindi distinto attraverso una definizione di questo tipo:

```
_<Protocol Name>._<Host-to-Host Transport Protocol>
```

Dove con "Protocol Name" si intende il nome ufficiale definito da IANA del protocollo (ftp, http, ecc) e con "Host-to-Host Protocol" si definisce il tipo di protocollo di trasporto che il servizio usa (tcp oppure udp).

```
_http._tcp.local
```

Nel caso si voglia definire un nuovo servizio è necessario in primo luogo registrarlo presso il sito web di IANA (<http://www.iana.org>).

Di seguito è riportato il grafo dei domini relativo alla denominazione dei servizi in Rendezvous. Partendo dalla radice che evidenzia attraverso il suffisso *.local* l'uso di domini di tipo “locale” passiamo attraverso la definizione del protocollo di trasporto, della tipologia di servizio sino a giungere nel livello più basso al nome assegnato per un determinato servizio evitando all'utente di ricordare stringhe poco “user-friendly”. Per tale nome possono essere usati una serie di caratteri unicode (UTF-8) qualsiasi purchè maggiore di 63 bytes.

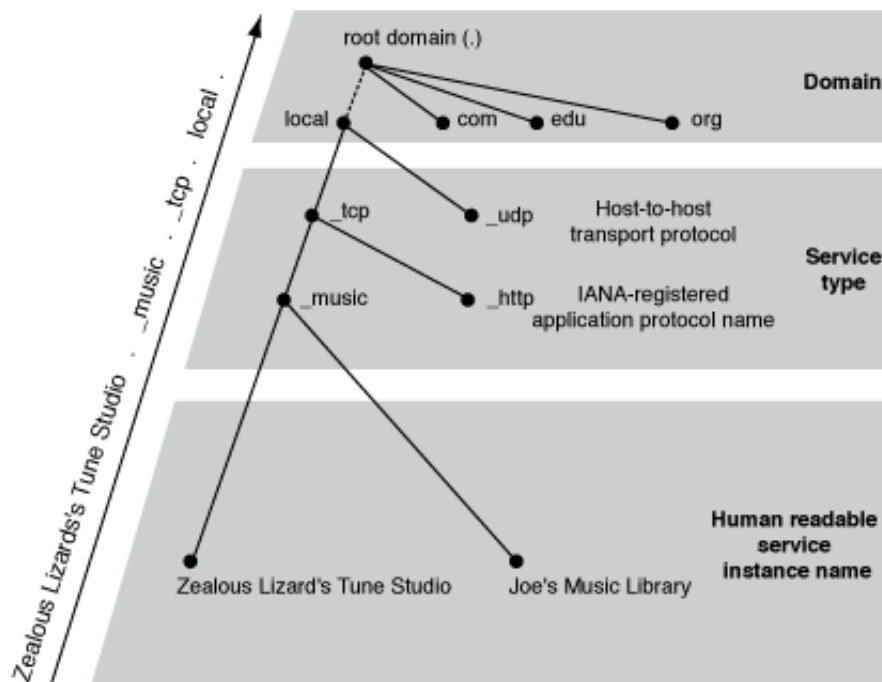


Figura 3: Grafo dei domini in Rendezvous

3.2 Rendezvous Service Architecture

Per permettere la fruizione di un servizio è necessario fornire tre meccanismi principali:

- *Publication:* (Pubblicazione di un servizio)

- *Discovery*: (Ricerca e Selezione di tutti i servizi disponibili)
- *Resolution*: (Traduzione dei nomi dei servizi in indirizzi e porte)

3.2.1 Publication

Attraverso la pubblicazione, un host mette a disposizione di tutti il proprio servizio. Per fare questo deve registrarlo attraverso un multicast DNS responder. Questa operazione può essere compiuta o attraverso l'uso di API di alto livello o attraverso la comunicazione diretta con il responder³. Una volta registrato il servizio, nel responder sono creati due record: uno contenente il service record (SRV) e l'altro contenente un puntatore alla tipologia di servizio (PTR). Esiste inoltre un ulteriore campo (opzionale) che contiene ulteriori informazioni sul servizio (TXT).

Il record di SRV è quello che effettivamente mappa il nome del servizio con le informazioni necessarie per la connessione e l'uso. Questo meccanismo permette facili interrogazioni al DNS, e non comporta problemi nel caso di eventuali modifiche ai dispositivi che implementano il servizio (ip o porte diversi, host name diverso, ecc) Nel dettaglio in SRV sono contenuti:

HostName: Nome della macchina sulla quale risiede il servizio.

Port Number: Identifica la porta tcp o udp per accedere a tale servizio

Priority: Funzionalità attualmente non implementata.

Weight: Funzionalità attualmente non implementata.

Un esempio di record SRV è il seguente:

```
myWebSite._http._tcp.local. 60 IN SRV 0 0 80 Mojito.local
```

che identifica un server web (http, tcp) disponibile nell'host denominato Mojito sulla porta 80.

Il record di PTR fornisce invece un'associazione tra una tipologia di servizio e un elenco di nomi delle specifiche istanze di quel servizio che sono attualmente disponibili.

³Vedremo meglio questa parte nella sezione dedicata agli sviluppatori.

Il TXT record aggiunge ulteriori informazioni riguardanti un determinato servizio. L'associazione record-servizio viene fatta sulla base dell'identificativo del servizio usato nel SRV record. Queste informazioni che non possono essere maggiori di 100-200 bytes possono fornire dettagli specifici sul determinato servizio (nel caso di un gioco multiplayer potrebbe contenere l'indicazione di quale mappa bisogna usare) Nel caso di utilizzo di servizi vecchi che facciano uso entrambi della medesima porta, questo campo può fornire un meccanismo di demultiplexing.

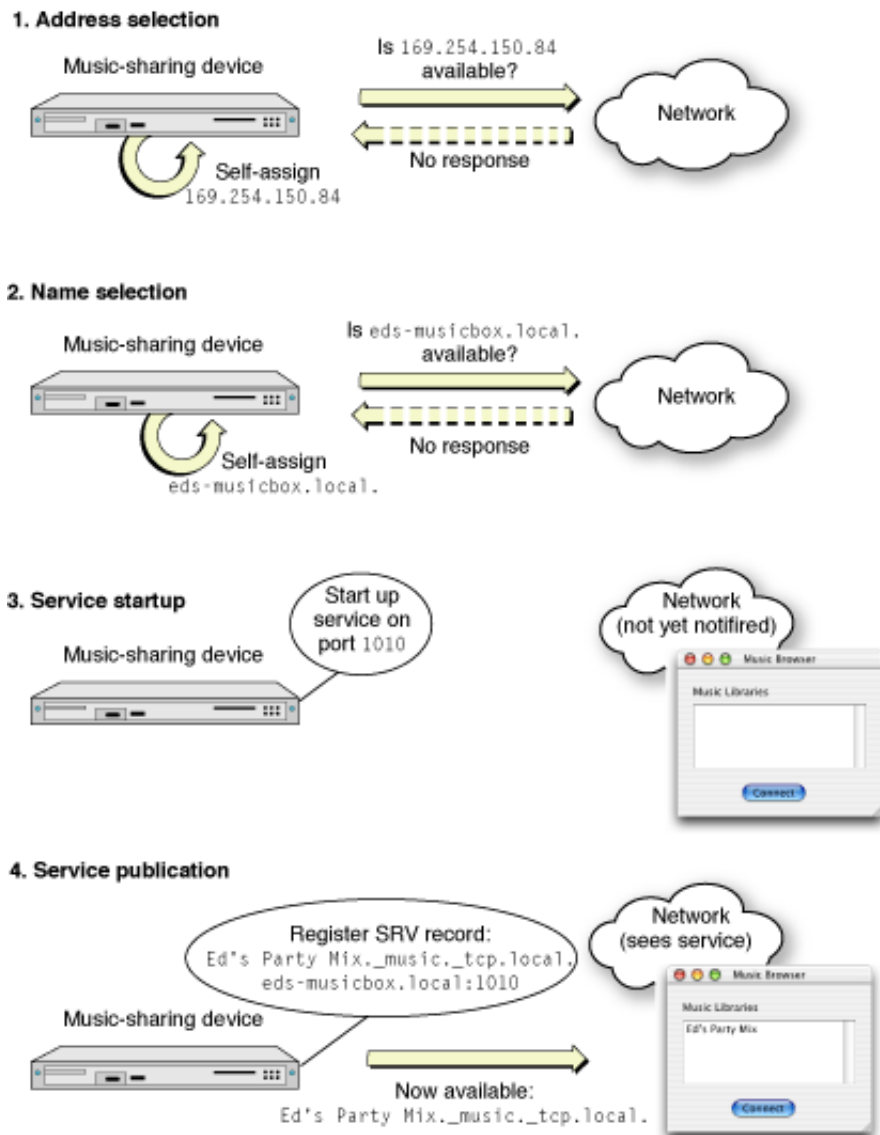


Figura 4: Presentazione di un servizio

3.2.2 Discovery

La ricerca di servizi con Rendezvous è molto semplice poichè avviene interrogando i DNS record registrati durante la fase di pubblicazione, in maniera da cercare tutti i nomi di servizio che corrispondono alla categoria richiesta. Questa query al record PTR avviene attraverso delle primitive API di alto livello.

3.2.3 Resolution

Resolution è il procedimento attraverso il quale un nome identificativo di un servizio viene trasformato in una informazione a livello di socket. Questa operazione si compone di due parti fondamentali:

1. Per svolgere questa traduzione l'applicazione svolge un DNS lookup per un particolare SRV record identificato attraverso il nome del servizio. Il record SRV corrispondente, contenente le informazioni necessarie, viene inviato al richiedente.
2. Una volta determinato l'host (attraverso il dominio locale) è necessario convertire tale nome con l'indirizzo ip della macchina che effettivamente detiene il servizio. Questa operazione viene compiuta attraverso una richiesta in multicast e viene effettuata tutte le volte che si desidera utilizzare il servizio poichè bisogna essere certi delle informazioni di socket attuali della macchina servente.

E' da notare che la prima operazione, a differenza di questa viene eseguita solamente nel fase successiva ad ogni discovery.

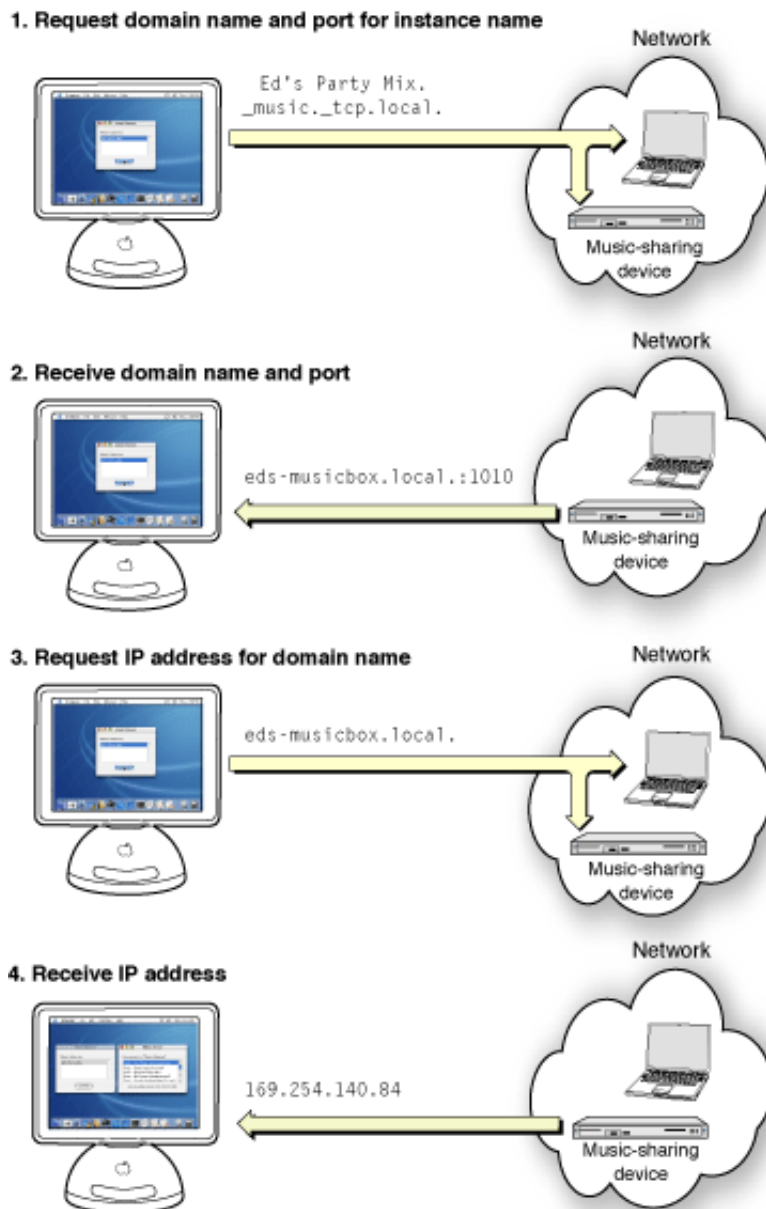


Figura 5: Traduzione del .local name.

4 Rendezvous per lo sviluppatore

Precedentemente abbiamo analizzato gli aspetti architetturali della tecnologia Rendezvous, cercando però di mantenere un'astrazione generale rispetto agli aspetti puramente implementativi che entrano in gioco pensando di realizzare applicazioni che utilizzino queste caratteristiche.

Considereremo ora le API a disposizione degli sviluppatori per integrare queste funzionalità in maniera semplice nelle loro applicazioni. D'ora in poi tratteremo in maniera dettagliata solamente l'implementazione delle API Rendezvous fornite con Mac OS X 10.2 e successivi. Esistono però implementazioni alternative di questa tecnologia che mettono di fatto a disposizione di tutti i sistemi le stesse caratteristiche. E' importante sottolineare che questa tecnologia non ha predisposizione solamente per Mac OS X, ed è appunto questo il punto di forza, anche in considerazione alla varietà degli hosts nelle reti attuali. Citiamo *Howl*⁴ come la principale implementazione open-source cross-platform. Queste API sono disponibili per Windows 2000/XP e Linux/FreeBSD.

Mac OS X fornisce tre livelli di Application Programming Interfaces (APIs) per la gestione dei servizi Rendezvous. Alla base di questa architettura risiede il Multicast DNS Responder che si occupa di gestire a basso livello e a stretto contatto con il sistema operativo tutte le richieste provenienti dagli strati superiori.

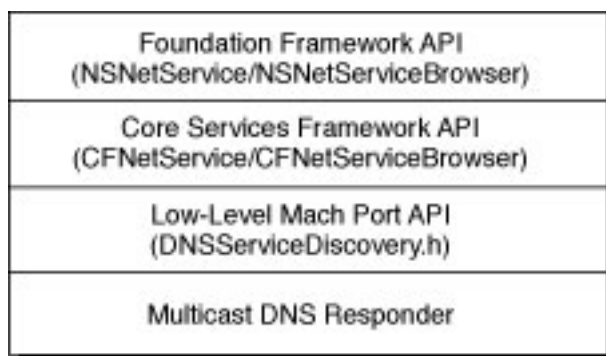


Figura 6: API Layers

⁴<http://www.porchdogsoft.com/products/howl/>

4.1 NSNetService e NSServiceBrowser

Le classi appartenenti a questi due livelli, facenti parte del Foundation framework Cocoa forniscono un'astrazione ad oggetti per il discovery e la pubblicazione di servizi. In particolare un oggetto di tipo NSNetService rappresenta un'istanza di servizio Rendezvous mentre un oggetto NSNetServiceBrowser rappresenta la ricerca di un particolare servizio all'interno di un particolare dominio. Entrambe le classi operano in maniera asincrona, e ritornano oggetti delegati. L'uso di delegati rappresenta un design pattern nel quale un oggetto compie operazioni per conto di un'altro oggetto. La relazione tra membro delegato e proprietario è realizzata tramite il concetto di "create"

L'esecuzione in maniera asincrona vuole evitare il problema dovuto alla lentezza di esecuzione dei processi (nelle reti i tempi di latenza sono altissimi) evitando quindi che il processo che ha inviato informazioni in rete e che impiega un certo tempo venga fermato con la conseguente necessità che lo stesso debba successivamente rinviare la richiesta.

NSNetService rappresenta un servizio che può essere remoto nel caso sia l'applicazione che deve utilizzare determinati servizi oppure può essere un servizio locale nel caso sia il software che deve pubblicarne uno. Queste due modalità sono però gestite in maniera esclusiva.

NSNetServiceBrowser può indicare la ricerca legato ad un particolare servizio o a un particolare dominio, ma in entrambi i casi un oggetto di questo tipo può riferirsi solamente ad una operazione di ricerca. Più ricerche implicano più oggetti di questo tipo.

Riassumiamo quindi i compiti delle due classi:

NSNetService: Creazione e pubblicazione di un servizio; Risolutore di indirizzi (dopo aver determinato il nome dell'host che ospita il servizio è necessario tradurre tale informazione in un indirizzo di rete valido)

NSServiceBrowser: Ricerca di host che forniscono servizi.

Spesso queste due classi forniscono tutte le potenzialità richieste dallo sviluppatore, mantenendo alto il livello di astrazione.

4.2 CFNetServices

CFNetService contiene tutte le funzioni e i tipi per gestire servizi e il discovery degli stessi. Sono della API a più basso livello, e sono le librerie su cui sono state definite le classi del layer superiore. Si prestano quindi per l'utilizzo all'interno di programmi C e C++.

4.3 DNSServiceDiscovery

Scendendo ancora di livello incontriamo questa classe di librerie. Esse forniscono una comunicazione a livello di kernel per l'uso di Rendezvous, poichè interagiscono direttamente col DNS responder. Usare queste librerie invece che accedere direttamente al responder permette di ragionare ancora in termini di servizi piuttosto che di records di risorse. Tali API andrebbero usati durante lo sviluppo di applicazione destinate al kernel (Darwin)

4.4 Un semplice esempio:Itunes Playlist Browser

Per mostrare in breve le potenzialità di Rendezvous analizziamo ora una semplice applicazione in C, che utilizzi le API presenti nella Core Foundation.

Il codice sorgente dell'esempio è fornito da Apple e può essere recuperato nel package Developer in /Developer/Examples/Networking/main.c

La nostra applicazione dovrà semplicemente ricercare all'interno della rete locale tutti i servizi della tipologia:

```
_daap._tcp.
```

L'applicazione è composta da tre funzioni principali e dal main, ognuna di queste tre funzioni assolve un compito ben preciso legato al discovery del servizio. Ricordiamo, prima di presentare i metodi, che la ricerca di un servizio deve essere associata ad un dominio e benchè tutt'ora l'unico dominio "usabile" per Rendezvous risulta quello definito con il suffisso *.local*, l'applicazione si deve fare carico anche della ricerca dei domini disponibili.

BrowseForServers: Questa funzione si occupa di gestire la ricerca, creando un network service browser object per ogni dominio

DomainBrowserCallBack: Poichè esisteranno più browser (uno per dominio e per servizio), DomainBrowserCallBack si preoccupa di gestire la ricerca di servizi invocando un service browser per ognuno di questi.

ServiceBrowserCallBack: Recupera da un service browser le informazioni legate al servizio e le stampa a video.

L'idea quindi è: Aggiungiamo un record per ogni dominio, e su ognuno di questi andiamo a ricercare i servizi. Il codice dell'applicazione non è di immediata comprensione, ma dobbiamo ricordarci che stiamo usando API di basso livello. Le stesse operazioni usando Objective C o Java sarebbero state decisamente più semplici.

Per provare e modificare la nostra applicazione useremo Xcode, il potente strumento di sviluppo di Mac OS X 10.3⁵.

Poiché Apple fornisce anche il file di progetto basterebbe cliccare su tale file con estensione `.pbproj` per poter provare e modificare la semplice applicazione. Per questo esempio si è deciso di creare il progetto da zero.

- Si lanci l'applicazione Xcode, presente in */Developer/Applications*.
- Cliccare su "File" --> "New Project"
- Selezionare "Empty Project"
- Immettere un nome per il progetto (Per esempio: rendezvous)
- Cliccando e premendo contemporaneamente il tasto "ctrl" (Tasto destro) sul nome del progetto, selezionare "Add Existing Files". Selezionare il file sorgente *main.c* dell'esempio e premere "Add".
- Importiamo ora il framework necessario: Come prima usando il tasto destro sul nome del progetto "Add Existing Frameworks". Selezionare "CoreServices.Framework".
- Definiamo il target file: Dal menù "Project" --> "New Target...". Selezionare "Kernel Extension Tool" e inserire un nome per il target file. Premere "Finish"

⁵Se non presente nel vostro sistema, è necessario installare il Developer Package presente nel quarto cd del sistema operativo.

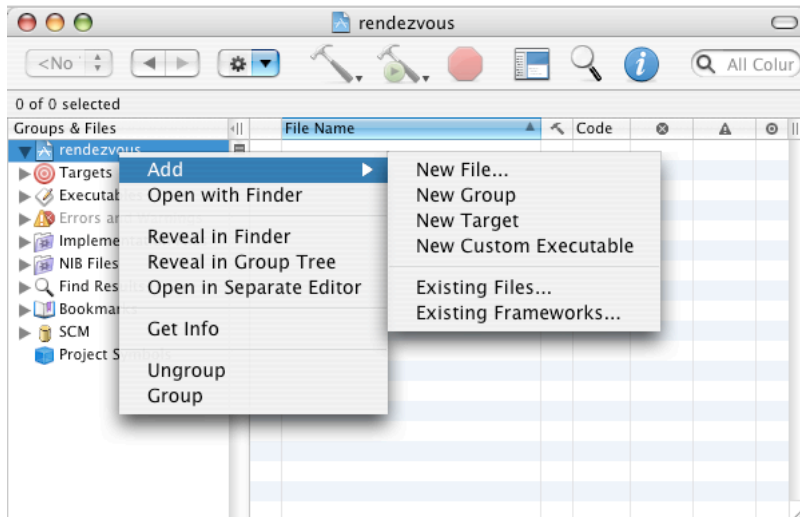


Figura 7: Aggiunta del file d'esempio nel progetto

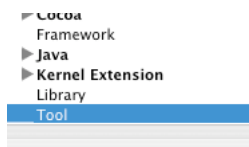


Figura 8: Aggiunta del target file

- In ultimo: “Project”-->“Edit Active Target”. Selezionare dalla finestra dei files del progetto il file C dell’esempio e trascinarlo all’interno della sezione “Build Phases”-->“Source”. Eseguire la medesima operazione trascinando anche il framework nella sezione “Build Phases”-->“Frameworks&Library”. Per maggiori dettagli vedere la figura numero 9.

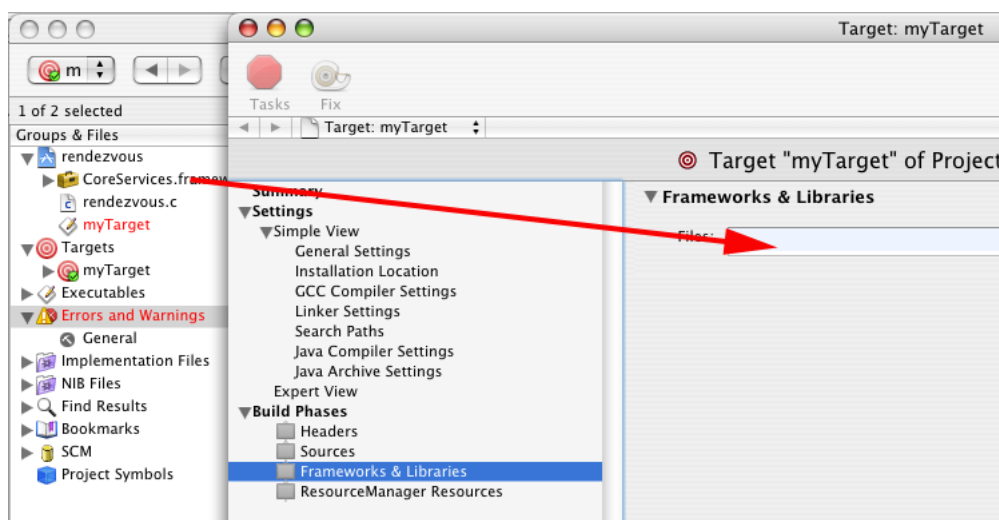


Figura 9: Editare il target file

- Chiudere la finestra e lanciare l’applicazione.

È possibile osservare che l'applicazione effettua il browsing per quel particolare servizio, e rivela (poichè in esecuzione iTunes sulla medesima macchina) una playlist condivisa.



Figura 10: Ricerca del servizio di music sharing di iTunes