Alma Mater Studiorum – Università Di Bologna

FACOLTA' DI SCIENZE MATEMATICHE FISICHE E NATURALI

Corso di Laurea Triennale in Informatica

CONNESSIONE A BASE BLUETOOTH

Tesi di Laurea in Architettura degli Elaboratori

Candidato:

Marco Tiseo

Relatore:

Prof. Vittorio Ghini

Sessione II

Anno Accademico 2010-2011

Capitolo 1

Introduzione

Ringrazio i miei genitori che mi hanno permesso di seguire questo corso di laurea e la mia ragazza Chiara che mi è stata sempre vicino.

La diffusione di terminali mobili avvenuto in questi anni ha contribuito di gran lunga a migliorare le nostre condizioni di vita. Basti pensare alle varie applicazioni che questi dispositivi ci mettono a disposizione quali navigatori satellitari applicazioni di messaggistica, social network, applicazioni d' ufficio che possono essere eseguiti anche online.

Su questi terminali spesso è proprio la connettività che la fa da padrone come citato in precedenza, è quindi impossibile pensare a un dispositivo mobile isolato dal mondo.

Con questa tesi si cerca di risolvere il problema della connettività da un punto di vista pratico e intuitivo.

Lo scenario in cui pensiamo di realizzare il nostro supporto alla mobilità è uno scenario prettamente urbano dove con il passare degli anni si pensa sempre più alla condivisione alla connettività.

Le città moderne mettono a disposizione connessioni gratuite i musei offrono connessioni per essere visitati in modo interattivo e le persone scelgono di telefonare utilizzando il VOIP.

Questi sono solo alcuni dei servizi di cui ogni cittadino munito di un device mobile di ultima generazione può usufruire.

E per questo motivo che nasce l' ABPS (Always Best Packet Switching) questo sistema si prefigge lo scopo di rendere le comunicazione voip sempre a disposizione in tutti gli scenari sfruttando tutte le interfacce di rete a disposizione sul device in modo da limitare consumi di tipo energetico ed economico.

Il documento che segue è formato da otto capitoli compreso l' introduzione, nel primo capitolo troviamo una breve introduzione nel secondo si passerà all'analisi dello scenario , nel terzo verrà presentato il protocollo ABPS.

Dal 4 in poi si entrerà un po nello specifico iniziando a parlare del protocollo bluetooth

della storia, delle caratteristiche tecniche . Il quinto Capitolo invece prende in esame gli obiettivi che ci siamo prefissi di raggiungere in questa tesi. Il sesto descrive gli strumenti utilizzati mentre il settimo utilizza gli strumenti descritti per realizzare il lavoro vero e proprio.
Il documento termina con l'ottavo capito nel quale ci sarà una breve conclusione .

Capitolo 2

Scenario

Lo scenario a cui fa riferimento il modello ABPS ha come oggetto principale la comunicazione VoIP su un dispositivo mobile, che può essere un laptop o con maggiore probabilità un dispositivo di telefonia mobile, come nel caso oggetto di tesi, equipaggiato con più di un'interfaccia di rete (NIC) wireless come WiMax, WiFi (IEEE802.11/a/b/g/n), GPRS, EDGE, 1xRTT, ZigBee, Bluetooth o altre .

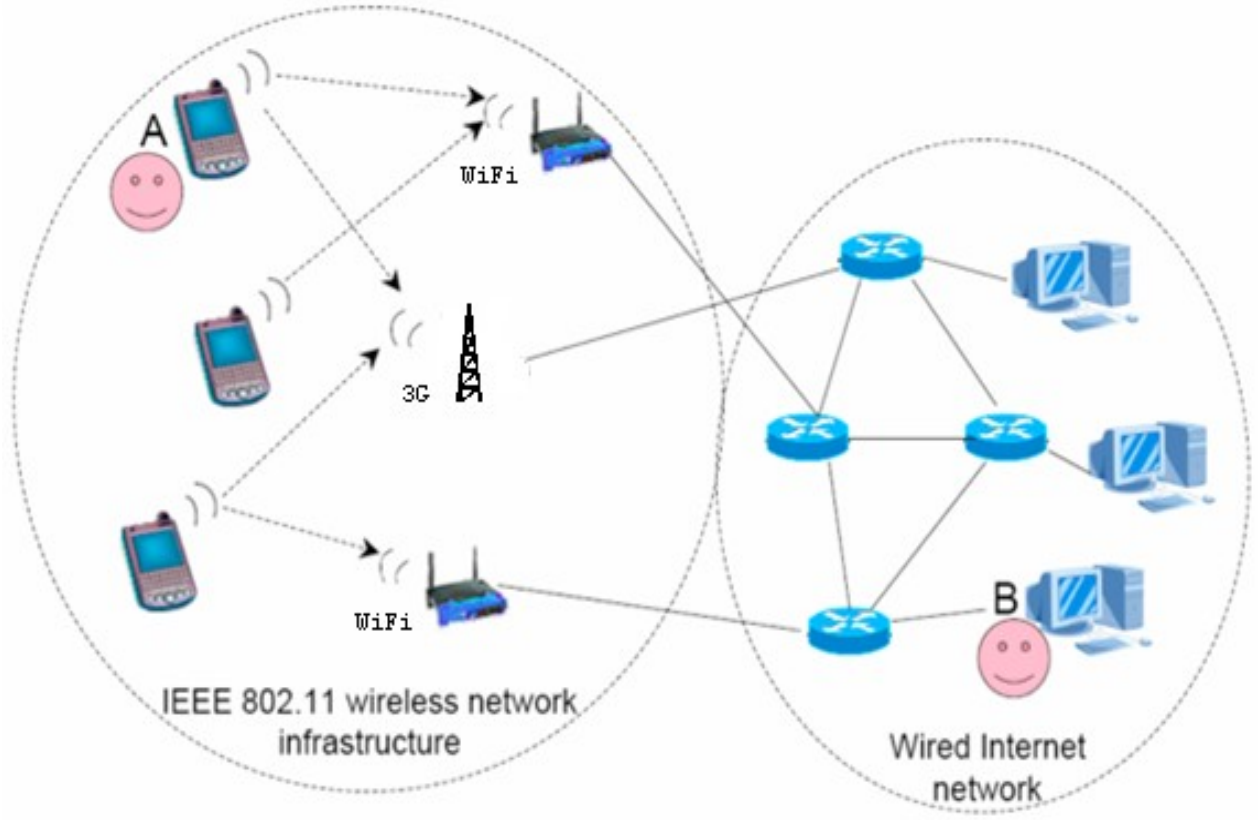


Figura 1 Scenario Multihoming handover

La figura 1 illustra lo scenario di due terminali VoIP A e B, di cui il primo mobile, equipaggiato con due o più interfacce wireless, una Wi-Fi in standard 802.11b/g/n, una 3G, posizionato in una tipica area metropolitana che offre sia copertura 3G che WiFi. Il terminale B è collegato direttamente ad internet mediante un collegamento wired.

Il vantaggio di avere a disposizione interfacce di rete eterogenee sullo stesso dispositivo mobile è evidente soprattutto nel caso di interfacce wireless, perché rende possibile la continua connettività nel caso venga meno una delle connessioni, estendendo la copertura delle aree WiFi con la copertura GPRS/UMTS. Una situazione del genere può verificarsi spesso in ambienti urbani, dove un utente si sposta in diversi punti della città cambiando access point e interfacce di rete, e viene definita dalle specifiche 3GPP come “Voice Call Continuity” (VCC). Un altro vantaggio è la selezione, nel caso sia disponibile più di un’interfaccia, del NIC preferito per accedere ad Internet. Quest’ultimo approccio è utilizzato dalla funzione di Mobility Management delle reti wireless, seguendo il modello di “Always Best Connected” (ABC), che suggerisce appunto di selezionare la migliore interfaccia NIC da usare come singolo punto di accesso a Internet. Quando le prestazioni di un’interfaccia degradano eccessivamente, il dispositivo mobile rileva una nuova interfaccia NIC preferita e la sostituisce alla prima.
L’architettura Always Best Packet Switching si basa su questo modello, ponendosi come scopo quello di offrire all’utente mobile il massimo della qualità di comunicazione sfruttando al meglio le capacità aggregate di tutte le interfacce di rete disponibili.
A causa di limitazioni tecniche ed economiche l’utilizzo di servizi multimediali come le conferenze audio/video su Internet mediante i dispositivi precedentemente descritti presentano ancora problematiche irrisolte, nonostante le continue innovazioni in campo, come connessioni più veloci e multihoming. Nel dettaglio, le applicazioni multimediali VoIP e Video on Demand (VoD) risentono maggiormente di queste limitazioni tecniche perché richiedono esigenze di QoS restrittive per poter offrire all’utente un buon livello di QoE. Per esempio, le raccomandazioni T G1010 stabiliscono che l’utente VoIP necessiti, per ottenere un servizio soddisfacente, di una latenza end-to-end minore di 150ms e una percentuale di pacchetti persi inferiore al 3%. Lo scenario VoWIFI quindi non permette ancora agli utenti un servizio affidabile e di qualità.

2.1 Il protocollo SIP

In breve, il protocollo SIP è un protocollo per creare, aggiornare e terminare sessioni multimediali, come chiamate vocali, tra due o più partecipanti. Il suo modello operazionale si ispira molto al mondo delle email, dove gli utenti sono identificati senza ambiguità dai loro indirizzi, ad esempio tiseo@cs.unibo.it, composti da un nome utente e da un dominio al quale l’utente appartiene.
SIP fa capo ad una topologia di rete astratta che viene spesso definita “SIP Trapezoid”, per motivi facilmente intuibili dalla figura 2. In questo trapezio gli angoli inferiori rappresentano i dispositivi

finali degli utenti, ad ognuno dei quali è associato un indirizzo simile a quello email. Nel lato basso viene trasmesso, usando il protocollo RTP, il flusso voce, con una linea diretta da un end all'altro. Prima che possa avvenire questa connessione diretta per lo scambio dei pacchetti RTP i due telefoni VoIP devono trovare i corrispettivi indirizzi IP e scegliere una codifica comune dei dati che entrambi possano capire per avviare la sessione audio, ed è proprio questo il compito del protocollo di signaling, il SIP. Le richieste di signaling partendo da ciascun terminale si propagano attraverso la rete per trovare la loro destinazione finale, e la parte superiore del trapezio rappresenta il percorso di questi pacchetti.

Questa divisione tra protocollo per signaling (SIP) e protocollo per il trasporto dei reali dati di applicazione (RTP) fa sì che mentre i pacchetti di signaling debbano attraversare il globo per raggiungere il server SIP di un certo dominio, i pacchetti voce, che sono in rapporto molti di più e costituiscono la vera comunicazione, vengono trasmessi utilizzando il percorso più breve possibile tra i due terminali. Prendendo come esempio la figura 2.2, se i due telefoni VoIP si trovassero nella stessa LAN, i pacchetti voce RTP non sarebbero trasmessi al di fuori della rete locale, mentre i pacchetti SIP dovrebbero comunque attraversare la rete Internet per raggiungere i server giga.com e mega.com.

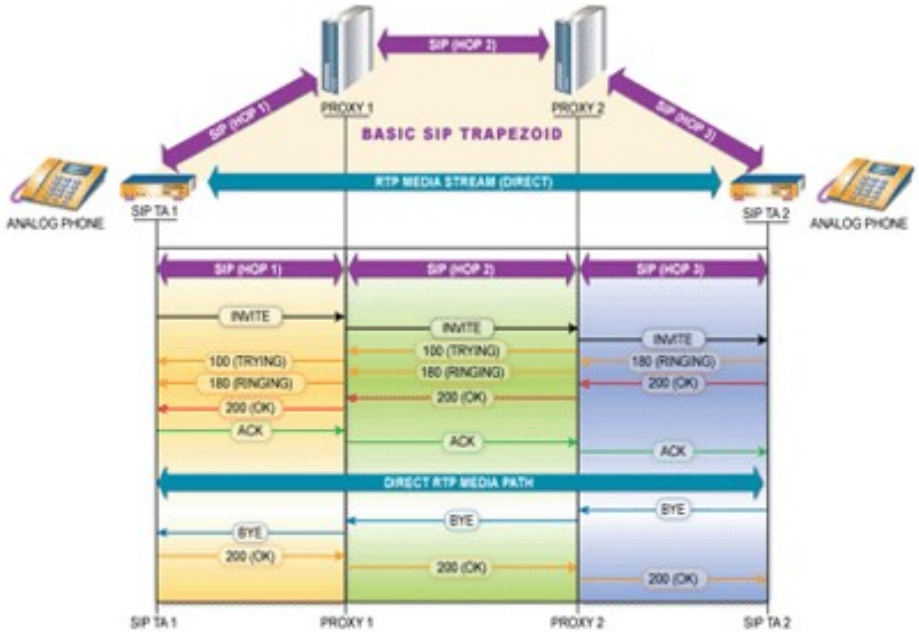


Figura 2. Il trapezio SIP

Per creare una sessione SIP sono necessari svariati componenti di rete. Nel caso più semplice possono essere necessari anche solo due telefoni SIP, ma nel caso più complesso di un servizio pubblico di telefonia SIP troviamo molti componenti di diversa da natura, come un insieme di

diversi SIP server per l'accounting, gateway PSTN per trasformare chiamate VoIP in chiamate su rete telefonica pubblica e viceversa, server per le conferenze, server voicemail eccetera. Questi componenti sono composti da una combinazione di pochi componenti logici semplici, descritti in : SIP User Agent, SIP proxy server, SIP redirect server e SIP Registrar.

Il SIP User Agent è l'elemento base di una comunicazione SIP, e rappresenta la funzionalità di SIP signaling presente in ogni componente di una rete SIP, anche se spesso riferendosi a questo componente si intende un client SIP software o un telefono SIP. Il SIP User Agent è un elemento peer-to-peer che può sia iniziare una richiesta SIP che riceverla.

Il SIP Registrar è un User Agent Server che mette in atto il concetto fondamentale su cui si basa il protocollo SIP, ovvero il legame tra indirizzo IP dell'utente e indirizzo permanente email-like SIP, chiamato Address of Record (AoR). Queste associazioni sono memorizzate all'interno dell'user location database del registrar, database che viene aggiornato dagli User Agent mediante delle richieste REGISTER.

2.2 Stato dell'arte per la seamless mobility

Le architetture per l'integrazione di reti eterogenee, note con il nome di Seamless Host Mobility Architectures sono responsabili di identificare univocamente un nodo multi-homed (MN), permettere ad ogni MN di poter essere raggiunto dagli altri nodi con cui ha già effettuato una connessione e monitorare il livello di QoS dei canali per predire un handoff selezionando la rete migliore così da mantenere la Voice Call Continuity. Queste architetture non hanno una posizione ben definita all'interno del classico stack ISO/OSI, possono essere invece implementate ad ogni differente livello della pila, dal livello di data link a quello di applicazione.

Nelle comunicazioni basate sul protocollo IP, l'indirizzo IP ha il compito di identificare l'MN, in quanto lo distingue univocamente e rappresenta la destinazione raggiungibile nei pacchetti proveniente dagli altri nodi con cui comunica. Purtroppo, essendo l'MN un nodo mobile che seleziona una rete differente a ogni spostamento in un insieme di reti sovrapposte, esso riceverà per ogni rete un diverso IP, perdendo a ogni selezione l'identità precedente. Quando ciò accade, rappresenta un problema soprattutto per i suoi nodi corrispondenti (CN), in quanto non potranno contattare l'MN prima di essere a conoscenza del suo nuovo indirizzo IP. Ne risulterà quindi una discontinuità all'interno della comunicazione, in contrasto con i principi VCC.

La soluzione comune a tutte le architetture di Mobile Management (MMA), anche se implementate in strati ISO/OSI differenti, si basa su due semplici principi:

1. Definire un identificatore univoco per l'MN, indipendente alla provenienza dell'host.
2. Fornire un servizio di localizzazione,sempre raggiungibile dai CN, per mantenere un'associazione tra l'identificatore univoco dell'MN e il suo reale indirizzo di provenienza.

Il servizio di localizzazione è composto da un Location Registry (LR) attivo su un server con indirizzo IP fisso e pubblico, raggiungibile da qualsiasi host. Se il CN è a conoscenza dell'identificatore univoco dell'MN, sarà sufficiente mettersi in contatto con il Location Registry per recuperare l'indirizzo IP dell'MN e inizializzare o continuare una comunicazione diretta.

Le architetture dello strato di rete attualmente utilizzate per la seamless mobility sono le seguenti:

- Mobile IP versione 6 (MIP)
- Fast Handover Mobile IPv6 (FMIP), un'ottimizzazione della precedente
- Hierarchical Mobile IPv6 (HMIP)
- Proxy Mobile IPv6 (PMIP)

Tutte le architetture elencate aggiungono all'interno della rete alla quale appartiene l'MN un'entità, detta Home Agent, avente funzione di Location Registry. Queste architetture, essendo basate su IPv6, richiedono che entrambi gli end, MN e CN, abbiano capacità IPv6, e necessitano di infrastrutture di rete che supportino IPv6. Inoltre le specifiche MIPv6 non permettono l'utilizzo simultaneo di più NIC dell'MN. Per ogni MN può essere registrato l'indirizzo di un solo NIC nell'home agent, lasciando di fatto irrisolto il problema degli intertechnology handoff. In più, come dimostrato la latenza dell'handover risulta molto alta a causa di numerosi messaggi di autenticazione.

Per le architetture situate tra il livello di rete e il livello di trasporto, come il protocollo Host Identity Protocol (HIP) e il Location Independent Addressing for IPv6 (LIN6), il location registry è rappresentato da una funzione di mapping simile al DNS che opera come servizio esterno alla rete di provenienza dei nodi. L'approccio di queste architetture si basa sull'aggiunta, nell'MN e nel CN, di uno strato intermedio tra il livello di rete e quello di trasporto ed è proprio questo principio a renderlo non conveniente, in quanto se può essere ragionevole una modifica simile nel sistema MN, non lo è negli altri nodi partecipanti, che potrebbero anche essere dei nodi fissi non interessati a supportare la mobilità dell'MN.

Per il livello di trasporto l'approccio comune è molto differente. Le architetture per questo livello, come il Datagram Congestion Control Protocol (DCCP) , il Mobile Stream Control Trasport Protocol (m-SCTP) e l'estensione TCP-migrate, integrano le fuzioni di location registry all'interno dei terminali stessi, che hanno il compito di informare gli altri end a ogni cambio di indirizzo IP. Con questo approccio il problema della localizzazione resta se i due (o più) end perdono la loro configurazione IP nello stesso momento, diventando così reciprocamente irraggiungibili.

Infine, per le architetture a livello di sessione, come il protocollo Terminal Mobility Support

Protocol (TMSP), abbiamo che il principio base sta nell'utilizzo di un server SIP ausiliario, esterno alla rete di accesso, che ha la funzione di location registry per mappare alle URI degli utenti il loro indirizzo IP corrente. Ogni MN utilizza il protocollo SIP per inviare un messaggio REGISTER al server SIP per aggiornare la sua localizzazione. Quest'ultimo soluzione non risulta efficiente perché quando avviene una riconfigurazione nell'IP dell'MN, il tempo durante il quale l'MN comunica al server di aggiornare il proprio record, mediante un handshake con diverse fasi, introduce un delay inaccettabile, durante il quale l'MN interrompe la comunicazione con il CN, che non può continuare la trasmissione prima dell'arrivo del messaggio di signaling da parte del server.

2.3 Requisiti per un supporto completo

Viste le limitazioni delle soluzioni attualmente esistenti, ci si rende conto della necessità di una soluzione specifica. I requisiti essenziali per un completo supporto alla seamless mobility nello scenario appena descritto sono i seguenti:

- trasparenza a livello utente: il roaming deve essere concluso il più velocemente possibile. L'utente non deve notare interruzioni nella comunicazione; quando questo non risulti possibile è necessario ridurre al minimo tali interruzioni.
- trasparenza a livello rete: non deve essere richiesto esplicito supporto nelle varie reti di accesso. Queste devono solo garantire connettività su protocollo IP.
- compatibilità: la soluzione deve essere completamente compatibile con lo scenario preesistente, ovvero con relative entità e protocolli. In una comunicazione tra un terminale mobile ed uno fisso non deve essere richiesto supporto specifico da parte del terminale fisso. Questo quindi deve rimanere ignaro della mobilità del terminale corrispettivo.
- QoS: la mobilità del terminale MN deve essere gestita rispettando adeguatamente i requisiti di QoS di quest'ultimo.
- full-mobile: deve essere supportata la possibilità che entrambi i terminali in comunicazione siano mobili. Questi devono quindi essere in grado di proseguire una comunicazione indipendentemente dai rispettivi spostamenti.
- nat-friendly: la soluzione deve essere compatibile con la presenza di politiche di NAT sulle reti di accesso. Questa non deve quindi risultare di ostacolo alle preesistenti tecniche di nat-traversal.

Si nota che, in base al requisito *trasparenza a livello rete*, le reti di accesso sulle quali si sposta il terminale devono solo fornire connettività su protocollo IP. Non si vuole infatti porre alcun limite alle tipologie di roaming gestibili, in modo da fornire una mobilità completa. L'idea è quella di sfruttare il multihoming per fornire la continuità della comunicazione in piena mobilità, il tutto

gestendo opportunamente eventuali riconfigurazioni dell’indirizzo IP utilizzato.

Capitolo 3

Architettura ABPS

Il sistema ABPS è una soluzione completa ed efficace di QoS e terminal mobility per il VoIP wireless basato su SIP, ed è composta da due entità:

- sip mobility: un'entità realizzata a livello applicazione che gestisce le conseguenze di un handover layer-3 (ovvero la riconfigurazione dell'indirizzo IP), che rispetti i requisiti precedentemente elencati.
- vertical mobility: un'entità realizzata a livello data-link e network per monitorare lo stato di ogni interfaccia di rete presente sul dispositivo, gestendo in base ad opportune metriche e politiche la selezione dell'interfaccia di trasmissione.

La prima entità è realizzata mediante un server posto sulla rete pubblica, l'altra è una applicazione eseguita direttamente sul terminale mobile.

Il server rappresenta il punto d'ancora del terminale mobile, ogni comunicazione VoIP del terminale passa attraverso di esso, qualunque entità VoIP-SIP che debba comunicare con il terminale contatta il server stesso credendo che presso di questo si trovi effettivamente il terminale. In questo modo la mobilità del terminale viene gestita direttamente ed esclusivamente dal server, similmente a quanto fatto da un home agent in MobileIP.

In conseguenza di una scelta progettuale dovuta soprattutto a questioni pratiche, la versione corrente del sistema ABPS prevede che l'applicazione includa diversi livelli con diverse funzioni. L'idea originale era infatti quella di utilizzare un proxy server ABPS in locale (sul dispositivo MN) per aggiungere ai pacchetti SIP ed RTP gli header ABPS, interfacciandosi con il FS in modo del tutto trasparente all'applicazione client. In questa nuova implementazione invece viene integrato tutto in un'unica applicazione client specifica per ABPS, divisa in più livelli, ad ognuno dei quali è assegnato un compito specifico dell'architettura. In particolare l'applicazione completa è composta da un livello più basso di selezione dell'interfaccia di rete, per effettuare handover e load balancing, un secondo livello formato da un insieme di funzioni e strutture per effettuare autenticazione mutuale e firma dei pacchetti, livello che verrà descritto in questa tesi, e di un terzo e un quarto livello che riguardano il client SIP vero e proprio, ovvero l'utilizzo dei livelli sottostanti per trasmettere al proxy server FS e lo sviluppo della GUI di un *softphone* SIP.

Le due entità descritte, il client ABPS su MN e il proxy su FS creano un tunnel logico tra esse

permettendo la continuità della comunicazione in modo completamente trasparente, sia al terminale che alle altre entità in comunicazione con questo.

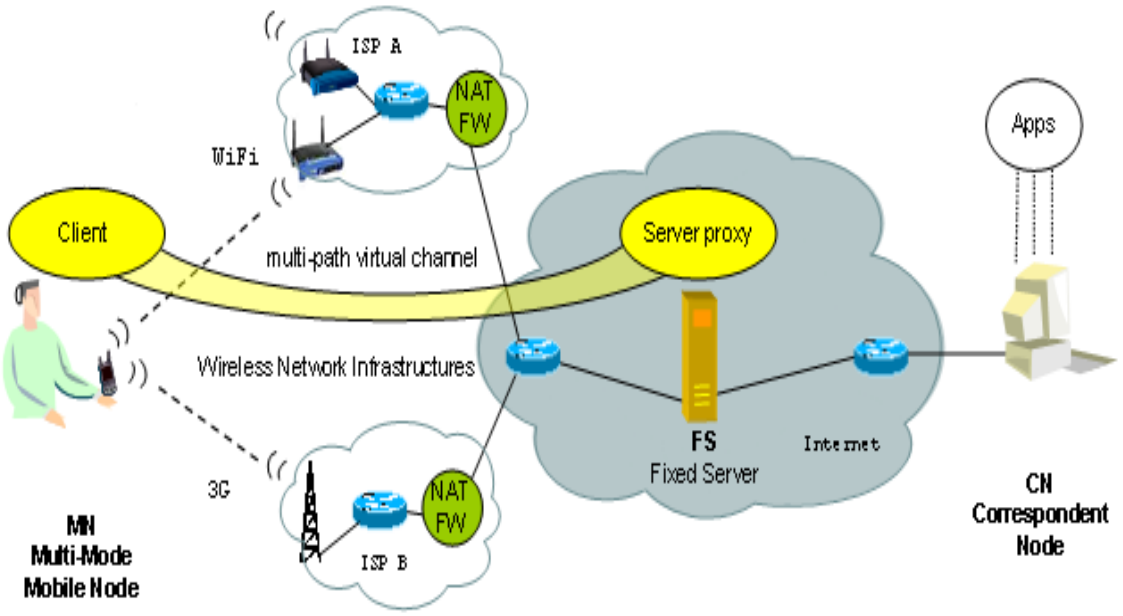


Figura 3. Scenario ABPS

In figura 3 è illustrato il nuovo scenario comprendente anche il sistema ABPS. Si noti come siano state introdotte le due entità:

- Client ABPS per il nodo mobile MN, che come descritto precedentemente è un'applicazione che estende le funzionalità di un classico UA offrendo un supporto completo alla seamless mobility. Viene eseguito direttamente sul terminale di Alice e svolge un ruolo simile al foreign agent di Mobile IP, ma contrariamente a questo non viene gestito dalla rete di accesso ma dallo stesso terminale.
- Server proxy ABPS sul fixed server (FS), è un proxy SIP che gestisce la mobilità in collaborazione con il client di Alice. Può essere visto come un home agent di Mobile IP, e si occupa di ricevere comunicazioni da e per il terminale mobile, inoltrandole opportunamente in base all'attuale posizione di questo. A differenza di un home agent non viene gestito dal provider di rete, può invece essere gestito dal provider SIP di Alice o da un provider terzo specializzato in questi servizi.

3.1 Sicurezza in ABPS

Essendo ABPS un sistema non *ip-centrico* è necessario che il server ABPS sia in grado di distinguere il traffico proveniente da un certo client registrato, senza potersi affidare all'indirizzo IP da cui questo proviene. Questo perché il client sul MN deve poter trasmettere i suoi dati da un qualunque indirizzo IP, anche da più indirizzi IP contemporaneamente, in un qualunque momento della sessione, senza dover avvisare il FS della riconfigurazione. Questo approccio fa sì che non sia necessario un protocollo di sincronizzazione, eliminando i relativi ritardi di trasmissione e mantenendo la continuità.

Per realizzare questo scenario il sistema ABPS si affida a due elementi:

- Comunicazioni crittografate: per ogni comunicazione (SIP ed RTP/RTCP) tra un MN e il FS si garantisce: autenticità, integrità ed opzionalmente (non ancora implementata) cifratura.
- Procedura di registrazione: ogni MN è identificato univocamente da un opportuno codice (MN-ID); al suo avvio il client, prima di ogni altra operazione, effettua una procedura di associazione con un opportuno FS. Tale procedura permette di creare un contesto di sicurezza, in particolare permette di stabilire opportune chiavi crittografiche per la sicurezza dei pacchetti SIP.

L'idea è quindi di creare un canale logico sicuro tra MN e FS, ovvero garantire la sicurezza dei protocolli con cui viene scambiato il traffico: SIP ed RTP/RTCP. Essendo le caratteristiche di questi due protocolli molto diverse sono state studiate due soluzioni separate e ottimizzate, con l'obiettivo primario di ridurre al minimo l'overhead introdotto con ABPS.

Capitolo 4

Bluetooth

4.1 Un po di storia

La storia di questo appellativo ci rimanda indietro nei secoli. Harald Bluetooth era il nome di un antico re vichingo, vissuto tra il 940 ed il 981 D.C. La particolarità di Harald Bluetooth fu l'unione di due paesi tanto diversi tra loro, come Danimarca e Norvegia, sotto un unico regno, con lui a capo. Inoltre Harald portò a termine la completa cristianizzazione dei due paesi. In tal modo il nome Bluetooth è rimasto a significare l'unione di due realtà diverse, ma con la volontà di collaborare per un futuro migliore. Così Bluetooth è stato preso come simbolo per la nuova tecnologia, che porterà ad unire differenti dispositivi, anche molto diversi tra loro, in un'unica grande rete senza fili. Anche il simbolo del Bluetooth rimanda all'antico re vichingo. Consiste, infatti, nei due simboli celtici raffiguranti la H e la B, le iniziali di Harald Bluetooth.
Le basi dell'odierno Bluetooth le getta Ericsson nel 1994. L'azienda svedese iniziò una ricerca per trovare una nuova interfaccia di comunicazione ad onde radio a basso costo, tra i cellulari ed i diversi accessori, che soppiantasse l'IRDA. L'idea era quella che un minuscolo ricevitore radio immesso sia nel cellulare sia nell'eventuale dispositivo da unire, avrebbero sostituito i vari fili usati all'epoca. Dopo circa un anno dal progetto iniziale, lo sviluppo della nuova tecnologia entrò nella fase operativa ed il lavoro passò alla sezione ingegneristica dell'Ericsson. Il progetto iniziale di fornire nuovi accessori per telefonia mobile ad onde radio, fu soppiantato da un progetto più ambizioso, la creazione di una vera e propria nuova tecnologia per far comunicare qualsiasi tipo di dispositivo a breve distanza. Ericsson capì l'importanza degli studi effettuati e decise di allargare il gruppo di sviluppo ad altri partner. Si arriva quindi al 1998, con la formazione del SIG (Special Interest Group) insieme a Nokia, Intel, IBM e Toshiba. Inizialmente Ericsson ha fornito un importante contributo per quanto riguarda la tecnologia radio. Toshiba ed IBM hanno lavorato per sviluppare un protocollo comune per l'integrazione del Bluetooth all'interno di dispositivi portatili. Nokia si è occupata della trasmissione dati e del software ed Intel della progettazione dei nuovi chip necessari. Gli obiettivi iniziali del gruppo erano aiutare lo sviluppo di una nuova tecnologia ad onde corte per far comunicare a breve distanza, creando uno standard fisso, ma aperto a tutte le aziende che ne volessero far parte. Già nel mese di aprile del 1999 il consorzio contava ben 600 membri. Oggi il SIG (Bluetooth Special Interest Group) è composta da più di 14.000 aziende associate (tra

cui citiamo Sony Ericsson, IBM, Intel, Toshiba, Nokia) che sono leader nel settore delle telecomunicazioni, informatica, elettronica di consumo, automotive, automazione industriale, e le industrie di rete.

Varie versioni del Bluetooth sono state ratificate dal SIG, tutte rispondenti ai requisiti di interoperabilità, armonizzazione della banda e promozione della tecnologia. Obiettivo principale del SIG è appunto garantire il perfetto funzionamento di apparati Bluetooth costruiti da differenti aziende. L'interoperabilità dei dispositivi Bluetooth è canone base per garantire il successo della nuova tecnologia, è quindi ovvio che sia il primo punto di interesse del SIG.

La tecnologia continua ad evolversi, a partire dalla sua forza intrinseca, il basso consumo, basso costo, la sicurezza, la robustezza, la facilità d'uso e la capacità di networking ad hoc. Le versioni sviluppate negli anni sono diverse l' ultima in ordine temporale è Bluetooth v4.0 questa versione unifica tutti i progressi evolutivi e fornisce ai produttori e ai consumatori un basso consumo energetico e le caratteristiche ad alta velocità per la connessione in modalità wireless in particolare il raggio di azione è stato esteso, passando da 10 a 60 metri e sono stati potenziati i meccanismi di rilevazione e correzione di errore e di criptatura del segnale col supporto di AES-128. Produttori e consumatori possono scegliere quali di queste caratteristiche attivare nei dispositivi che supportano la tecnologia Bluetooth, a seconda della funzionalità del dispositivo.

Figura 4 primo dispositivo bluetooth ericsson T36

4.2 Introduzione

Bluetooth è la tecnologia per la trasmissione delle informazioni tramite le onde radio, che serve a collegare fra di loro telefoni cellulari ed ogni genere di computer o periferica, senza l'ausilio di scomodi cavi di connessione. Permette l'utilizzo di interfacce radio che consentono di collegare fra di loro telefonini, computer ecc. Nel periodo in cui ericsson iniziò a sviluppare questa tecnologia iniziavano a nascere numerosi gadget per lo stesso device il che rendeva scomodo il loro utilizzo contemporaneo sopratutto in mobilità a causa della presenza di fili .
Con la tecnologia Bluetooth tale limitazione sparisce: le onde radio viaggiano tranquillamente anche attraverso ostacoli, come pareti o custodie di computer portatili, e consentono un utilizzo immediato di file e di informazioni. In pratica niente fili per collegare periferiche.
I dispositivi si connettono l'uno all'altro anche a decine di metri di distanza, in movimento, anche da una stanza all'altra. Le periferiche che possono essere collegate tramite Bluetooth sono moltissime. E' uno dei mezzi più importanti per lo scambio di dati tra terminali mobili. Il gioco in multiplayer è possibile anche off line, basta avere tutti il medesimo videogame sui propri terminali ed essere vicini qualche decina di metri per poter giocare . Per esempio i joyped possono tranquillamente essere collegati alla console tramite questa tecnologia rendendo il tutto più comodo e interattivo basti pensare alle nuove console come Nintendo Wii o Playstation 3.
Facilita sia la comunicazione voce che quella dati ed offre la possibilità di creare delle vere e proprie reti mobili apposite, sincronizzando i terminali con i dispositivi associati oppure creando una piccola rete domestica per lo scambio di dati tra terminali o ancora la condivisione della rete internet utilizzando un altro device con interfaccia bluetooth collegato in rete come gateway .
Questa tecnologia è talmente diffusa che più di otto nuovi prodotti vengono utilizzati tutti i giorni e oltre 30 milioni di unità vengono spedite ogni settimana . Ci sono oltre tre miliardi di dispositivi Bluetooth sul mercato, il che la rende l'unica scelta per gli sviluppatori, produttori e consumatori in tutto il mondo.

4.3 Informazioni Tecniche

La tecnologia Bluetooth è stata progettata per trasmettere dati a velocità superiori ad 1Mbps a distanze superiori a 10 metri. Un tale tipo di comunicazione permette il trasferimento di voce, di dati (incluse le immagini) attraverso diversi apparati Bluetooth. Ognuno di tali apparati può scambiare informazioni con un altro in connessione, per esempio dalla cuffia al proprio cellulare

Bluetooth è uno standard per reti PAN (Personal Area Network).

Una rete Pan è una rete informatica utilizzata per permettere la comunicazione tra diversi dispositivi (telefono, PC tascabile, ecc.) vicini tra loro, il suo raggio di azione e infatti di pochi metri .

Le PAN Bluetooth sono chiamate anche piconet e queste sono composte al massimo da otto dispositivi in relazione master slave . Il primo dispositivo Bluetooth è il master mentre i successivi diventano slave. Ogni dispositivo è in grado quindi di gestire simultaneamente la comunicazione con altri 7 dispositivi anche se essendo un collegamento di tipo master slave solo un dispositivo alla volta può comunicare con il server.

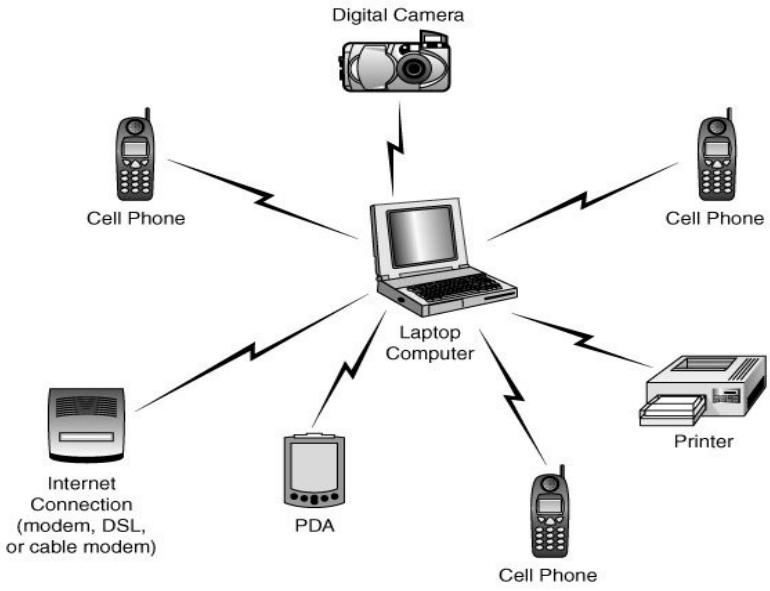


Immagine 5 piconet

Nella figura 5 è possibile notare una rete piconet formata da 7 dispositivi slave e da un dispositivo master che in questo caso è un laptop computer .

Una rete piconet ha un raggio tipico di 10 metri ma si possono collegare più piconet utilizzando un dispositivo che appartenendo a entrambe le piconet fa da ponte, in questo caso la rete che si viene a creare è chiamata scatternet .

Una scatternet è formata da più piconet interconnesse tra loro e supporta la comunicazione tra più di 8 dispositivi. Una Scatternet si forma quando un membro di una piconet (sia esso master o slave), sceglie di partecipare come slave in una seconda piconet . Il dispositivo che partecipa in entrambe le piconet può trasmettere i dati tra i membri di entrambe le reti . Usando questo approccio, è possibile unire numerose piconet in una scatternet .

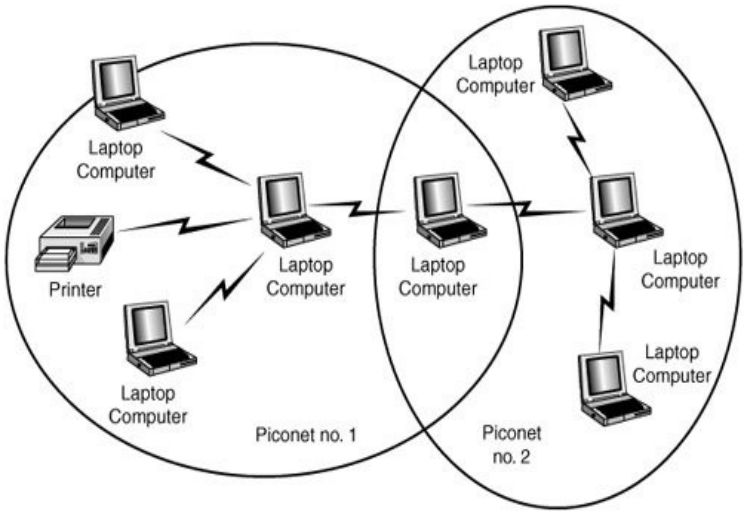


Immagine 6 scatternet

Nell'immagine (numero) potete notare una scatternet formata da 2 piconet nella quale il laptop computer in comune funge da master nella piconet numero 1 e da slave nella piconet numero 2.

Questa tecnologia opera nella banda ISM (Industrial, Scentific and Medical, un insieme di porzioni dello spettro elettromagnetico riservate alle applicazioni radio non commerciali, per uso industriale, scientifico e medico) dei 2,4 GHz suddivisa in 79 canali scanditi 1600 volte al secondo L'uso di queste bande può differire da stato a stato a causa di specifiche regolamentazioni nazionali .

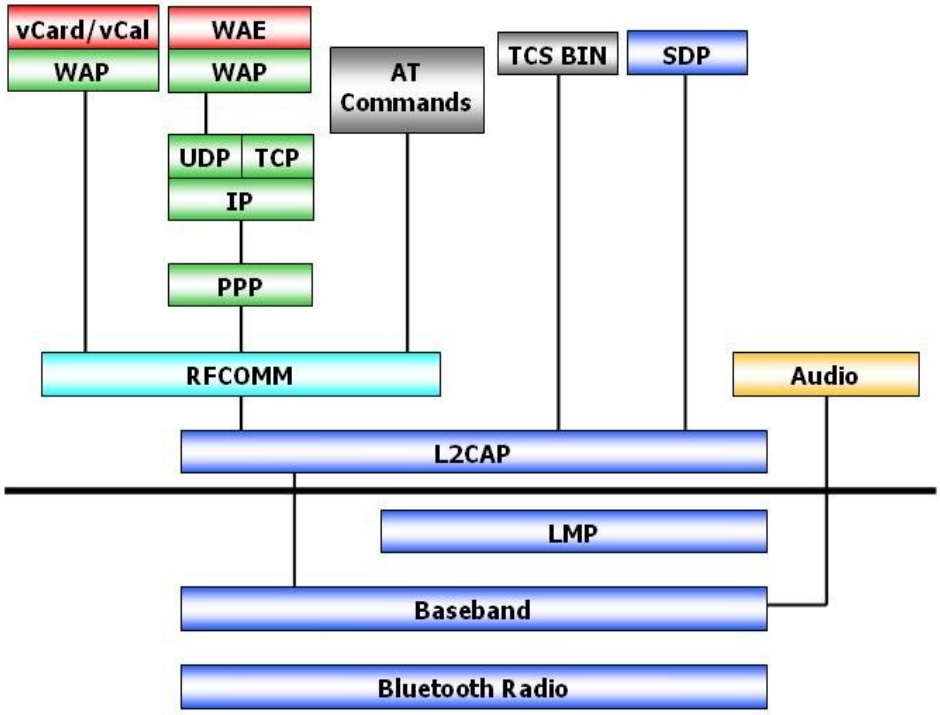
Country	**Frequency Range**	**RF Channels**
Europa & USA	2400 – 2483.5 MHz	f = 2402 + k MHz
Japan	2471 – 2497 MHz	f = 2473 + k MHz
Spain	2445 – 2475 MHz	f = 2449 + k MHz
France	2446.5 – 2483.5 MHz	f = 2454 +k MHz

La versione 1.1 e 1.2 del Bluetooth gestisce velocità di trasferimento fino a 723,1 kb/s. La versione 2.0 gestisce una modalità ad alta velocità che consente fino a 3 Mb/s. Questa modalità però aumenta la potenza assorbita. La nuova versione utilizza segnali più brevi, e quindi riesce a dimezzare la potenza richiesta rispetto al Bluetooth 1.2 (a parità di traffico inviato).

In relazione alla potenza invece i dispositivi si dividono in tre classi :

Classe	**Potenza (mW)**	**Potenza (dBm)**	**Range (m)**
Classe 1	100 mW	20 dBm	100 m
Classe 2	2,5 mW	4 dBm	10 m
Classe 3	1 mW	0 dBm	1 m

4.4 Stack Bluetooth



Fiugura numero 7 Stack bluetoot

Come per il modello ISO/OSI anche il Bluetooth è costituito da una pila (stack) di protocolli attraverso i quali è possibile ridurre la complessità di un sistema di comunicazione fornendo servizi sempre più complessi che permettono a dispositivi diversi di interagire tra di loro .

In particolare anche lo stack bluetooth è costituito da strati o layer , che racchiudono uno o più aspetti fra loro correlati come per esempio trovare dispositivi vicini, scoprire quali servizi offrono ecc.

Non tutte le applicazioni usano tutti i protocolli dello stak Bluetooth, infatti, esso è rappresentato su più livelli verticali, al di sopra dei quali c'è un'applicazione specifica.

Iniziando dai livelli più bassi e soffermandoci solo sui protocolli più rilevanti descriviamo brevemente lo stack:

Bluetooth Radio :

Si occupa dello scambio di informazioni in radio frequenza .

Baseband :

abilita il collegamento fisico tra dispositivi all'interno di una piconet. Tale livello si basa sulle procedure di inquiry che permette di fare uno scan e localizzare dispositivi presenti nell'area circostante ottenendo informazioni quali indirizzo bluetooth e clock del dispositivo remoto ,

di inquiry response per ritrasmettere al dispositivo scoperto informazioni utili sul proprio conto quali BD_ADDR, e di paging il quale permette di effettuare a tutti gli effetti un tentativo per stabilire una connessione attiva l'unità che stabilisce la connessione diviene così automaticamente il master.

LMP :

Il Link Manager Protocol effettua il collegamento, l' autenticazione, la configurazione del collegamento . Scopre altri LM remoti e comunica con loro tramite il Link Manager Protocol (LMP). Effettua anche il controllo sulle diverse modalità di gestione in cui il dispositivo si trova la modalità **Active mode** nella quale il dispositivo è abilitato sia alla trasmissione che alla ricezione, **Hold mode** modalità nella quale il dispositivo continua ad essere sincronizzato con il master ma non trasmette dati, **Sniff mode** in questo stato il device si trova in una modalità di risparmio energetico e **Park mode** modalità nella quale il dispositivo è ancora sincronizzato alla piconet ma perde il suo indirizzo di dispositivo attivo.

L2CAP :

Il Logical Link Control & Adaptation Protocol esegue il multiplexing dei protocolli di livello superiore, la segmentazione e il riassemblaggio dei pacchetti e il trasporto di informazione. L2CAP permette ai protocolli dei livelli superiori ed alle applicazioni di trasmettere e ricevere pacchetti di dati di dimensione superiore a 64Kbyte. Esso definisce solamente un collegamento di tipo connectionless.

RFCOMM :

Frequenza radio di comunicazione (RFCOMM) è un protocollo di sostituzione del cavo utilizzato per creare un flusso di dati seriali virtuali. RFCOMM fornisce meccanismi per il trasporto di dati binari ed emula una porta seriale EIA-232 (precedentemente RS-232) sul protocollo L2CAP. RFCOMM fornisce un semplice flusso di dati affidabile per l'utente, simile al TCP. E usato come livello di trasporto per OBEX un protocollo che facilita lo scambio di oggetti binari tra dispositivi .

SDP :

Service Discovery Protocol è un protocollo che mette a disposizione meccanismi per scoprire i sevizi offerti dai dispositivi con i quali si stabilisce una connessione .

TCS BIN:

opera a livello bit e definisce i segnali di controllo per le chiamate voce e dati tra dispositivi Bluetooth e le procedure per gestire gruppi di dispositivi TCS.

AUDIO :

La funzione di questo strato è quella di codificare il segnale audio. Due tecniche possono essere adottate: log PCM e CVSD; entrambe forniscono un flusso di bit a 64Kbps. La codifica log PCM (*Pulse Code Modulation*) consiste in una quantizzazione non uniforme a 8 bit. Nella codifica CVSD (*Continuous Variable Slope Delta Modulation*) il bit d'uscita indica se il valore predetto è maggiore o minore del valore della forma d'onda in ingresso, costituita da un segnale PCM con quantizzazione uniforme. Il passo è determinato dalla pendenza della forma d'onda.

PPP :

Nella tecnologia Bluetooth PPP è stato progettato per funzionare su RFCOMM per realizzare connessione punto-punto.
PPP è l'IETF Point-to-Point Protocol (Internet Engineering Task Force, Directory Lista di IETF RFC) e PPP-Networking è l' insieme di servizi per trasportare i pacchetti IP da e per il PPP Layer e inoltrarli sulla LAN.

TCP/UDP/IP :

Questi standard di protocollo sono già definiti dalla Internet Engineering Task Force e usati comunemente nella comunicazione via Internet.
Il protocollo TCP / IP è utilizzato in numerosi dispositivi come stampanti, computer palmari e telefoni cellulari.
L'uso del protocollo TCP / IP nel protocollo Bluetooth consente la comunicazione con qualsiasi altro dispositivo connesso ad Internet, per ottenere questo scenario però abbiamo bisogno di 2 dispositivi bluetooth uno collegato in rete e un altro che lo sfrutta come un ponte verso Internet.

OBEX

OBEX è un protocollo sessione sviluppato dalla Infrared Data Association (IrDA) per lo scambio di oggetti in modo semplice e spontaneo. OBEX fornisce le stesse funzionalità di base di HTTP, ma in maniera molto più leggera, utilizza un modello client-server ed è indipendente dal meccanismo di trasporto, a condizione che sia realizza una base affidabile di trasporto. Oltre alla "grammatica" per la conversazione tra dispositivi, OBEX fornisce anche un modello per la rappresentazione di oggetti e operazioni. Inoltre, il protocollo OBEX definisce meccanismi per visualizzare i contenuti di cartelle sul dispositivo remoto. OBEX usa RFCOMM o TCP/IP come livello di trasporto.

Capitolo 5

Obiettivi

Nel capitolo 2 abbiamo visto qual' è lo scenario di un modello ABPS (Always Best Packet Switching), riassumendo brevemente abbiamo un dispositivo mobile che ha più interfacce di rete e si muove in uno scenario urbano dove ci sono più acces point, in questo contesto L'architettura ABPS ha come scopo quello di offrire all'utente mobile il massimo della qualità di comunicazione sfruttando al meglio le capacità aggregate di tutte le interfacce di rete disponibili.

Questo dispositivo in movimento quindi può cambiare automaticamente interfaccia e acces point a seconda delle esigenze che possono essere di tipo economico oppure di tipo energetico oppure dovute a problemi relativi al mal funzionamento di una delle interfacce presenti o semplicemente legati alla mobilità .

Tra le interfacce di rete presenti su un dispositivo mobile quella che sicuramente non può mancare è l' interfaccia bluetooth la quale essendo molto economica permette di equipaggiare tutti i terminali sia di fascia bassa che alta .

Anche se la sua velocità di connessione è di gran lunga inferiore alle altre ha il pregio di essere molto economica e di mantenere i consumi energetici bassi, questo pregio si sposa a meraviglia con il modello ABPS il quale come abbiamo già spiegato si colloca in uno scenario di mobilità urbana dove il risparmio energetico ha sicuramente la sua importanza .

L' obiettivo che si prefigge questa tesi è quello di utilizzare un interfaccia bluetooth in modo da poter creare una piconet dove il dispositivo master è collegato ad una LAN e funge da NAP(Network Access Point) per i dispositivi slave che si collegano .

In questo modo un qualsiasi terminale sul quale è presente un interfaccia bluetooth è capace di limitare i consumi energetici e continuando ad offrire una discrete velocità di connessione.

Capitolo 6

Progettazione

Per poter realizzare l' obiettivo che ci siamo prefissi occorre andare alla ricerca degli strumenti hardware/software adatti al nostro scopo . Visto che la presenza di dispositivi adatti al modello ABPS e quindi con più interfacce di rete tra cui quella Bluetooth presenti sul mercato è illimitato, era improponibile pensare a una situazione generale che valesse per tutti poiché gli strumenti software che utilizzano differiscono in maniera sostanziale gli uni dagli altri .

Proprio per il fatto che ognuno di essi differisce in maniera sostanziale ho cercato di utilizzare strumenti adatti a tutte le tasche, e open source in modo da essere accessibile a tutti e facilmente modificabili .

Come dispositivi hardware quindi ho deciso di utilizzare due laptop computer con architettura Intel x86 sui quali ho installato una distribuzione Linux Ubuntu 11.10 .

Solo uno dei 2 computer è stato collegato alla rete mentre sul' altro ho disabilitato tutte le interfacce per focalizzarmi solo su quella bluetooth e verificarne l' effettivo funzionamento.

Ubuntu è un sistema operativo GNU/Linux nato nel 2004 questa distribuzione è stata rilasciata sotto licenza GNU GPL ed è gratuita e liberamente modificabile. La gestione dell'interfaccia bluetooth da parte di questa distribuzione è stata realizzata per mezzo di Bluez un software libero rilasciato anche esso sotto licenza GPL.

6.1 Bluez

Bluez è l' ufficiale linux bluetooth protocol stack ed è incluso in tutte le recenti distribuzioni. Fornisce supporto per i layer e per i protocolli bluetooth, è flessibile efficiente e utilizza un'implementazione modulare .

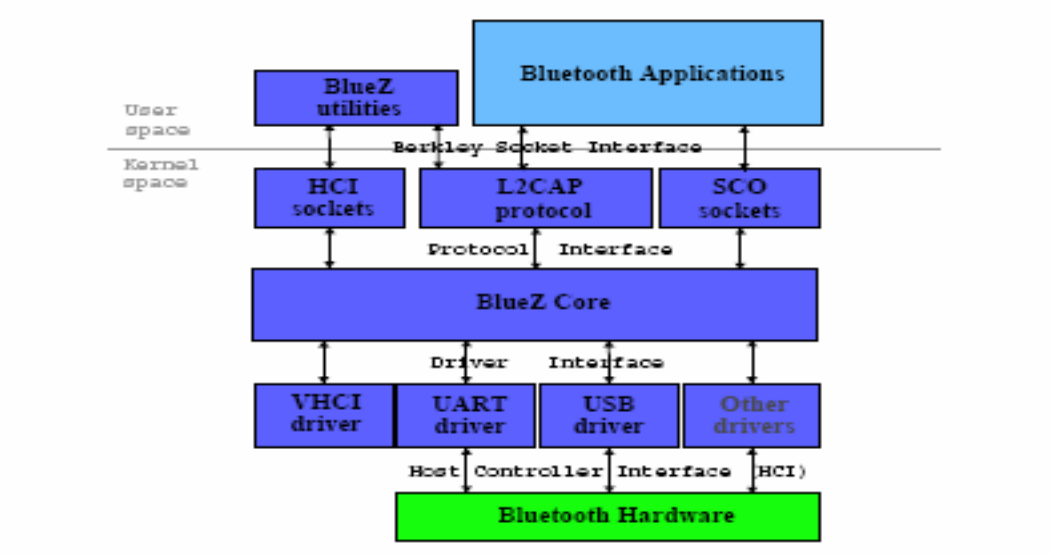


Figura numero 8 Diagramma bluez

Qui di seguito vengono riassunte le caratteristiche presenti nella figura (numero) :

* Implementazione completamente modulare
* Multi Processing sicuro
* Elaborazione dei dati multithread
* Supporto per più dispositivi Bluetooth
* Astrazione hardware
* Interfaccia socket standard per tutti i livelli
* Supporto per la sicurezza

Attualmente BlueZ consiste di molti moduli separati:.

* Bluetooth kernel subsystem core.
* L2CAP e SCO audio kernel layers.
* RFCOMM, BNEP, CMTP e le implementazioni del kernel HIDP.
* HCI UART, USB, PCMCIA e driver di periferica virtuale.
* Liebrerie Bluetooth, SDP e demoni.
* File di Configurazione e utilità di test.
* Protocollo di decodifica e strumenti di analisi.

I moduli del kernel, le librerie e le utility sono noti per lavorare perfettamente su molte architetture supportate da Linux :

* AMD64 and EM64T (x86-64)
* SUN SPARC 32/64bit
* PowerPC 32/64bit
* Intel StrongARM and XScale
* Hitachi/Renesas SH processors
* Motorola DragonBall

Il supporti che offre BlueZ si possono trovare in molte distribuzioni Linux e in generale è compatibile con qualsiasi sistema Linux sul mercato:

* Debian GNU / Linux
* Ubuntu Linux
* Fedora Core / Red Hat Linux
* OpenSuSE / SuSE Linux
* Mandrake Linux
* Gentoo Linux

6.2 Programmazione Bluetooth

Questo paragrafo presenta una panoramica della tecnologia Bluetooth, quella legata allo sviluppo e i concetti saranno spiegati nel quadro generale del protocollo TCP / IP e della programmazione di Internet.
Ogni chip Bluetooth ha un indirizzo a 48 bit, in maniera identica a un indirizzo MAC di Ethernet . Questi indirizzi sono assegnati al momento della produzione e sono destinati a essere unici e rimanere statici per tutta la durata del chip. Essi hanno molta importanza e sono alla base della programmazione bluetooth.
Se un dispositivo Bluetooth vuole comunicare con un altro, deve essere in grado di determinare l'indirizzo Bluetooth dell'altro dispositivo, questo verrà poi utilizzato a tutti i livelli del processo di comunicazione, dal più basso come il protocollo radio ai più alti come i protocolli applicativi. Al contrario invece nel protocollo TCP /IP rimane lo stesso il principio di rendere i dispositivi unici con l' uso di un indirizzo che però non e l' indirizzo mac del chip ma l' indirizzo IP.
Per rendere più familiari gli indirizzi di 48-bit come 0x000EED3D1829 questo sarà sempre associato a un nome tipo “MyDevice”. Questo nome è di solito indicato per semplicità dall'utente al posto dell'indirizzo , ma alla fine è l'indirizzo a 48-bit che viene utilizzato nella comunicazione reale. Per molte macchine, come telefoni cellulari e computer desktop, questo nome è configurabile e l'utente può scegliere una parola arbitraria o una frase.
Una volta che la nostra applicazione client ha determinato l'indirizzo della macchina host che vuole connettersi, è necessario determinare quale protocollo di trasporto debba essere utilizzato.
Il protocollo RFCOMM oppure il protocollo L2CAP.
Per illustrare un po il funzionamento della programmazione bluetooth ho deciso di mostrare alcuni esempi scritti in linguaggio C .
Vediamo ora un semplice programma scritto in linguaggio Ansi C che sta alla base della programmazione bluetooth, questo programma è in grado di rilevare i dispositivi nelle vicinanze stampando a video il nome associato all'indirizzo.

Scan.c

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/hci.h>
#include <bluetooth/hci_lib.h>

int main(int argc, char **argv)
{
    inquiry_info *ii = NULL;
    int max_rsp, num_rsp;
    int dev_id, sock, len, flags;
    int i;
    char addr[19] = { 0 };
    char name[248] = { 0 };

    dev_id = hci_get_route(NULL);
    sock = hci_open_dev( dev_id );
    if (dev_id < 0 || sock < 0) {
        perror("opening socket");
        exit(1);
    }

    len  = 8;
    max_rsp = 255;
    flags = IREQ_CACHE_FLUSH;
    ii = (inquiry_info*)malloc(max_rsp * sizeof(inquiry_info));

    num_rsp = hci_inquiry(dev_id, len, max_rsp, NULL, &ii, flags);
    if( num_rsp < 0 ) perror("hci_inquiry");

    for (i = 0; i < num_rsp; i++) {
        ba2str(&(ii+i)->bdaddr, addr);
        memset(name, 0, sizeof(name));
        if (hci_read_remote_name(sock, &(ii+i)->bdaddr,
          sizeof(name),name, 0) < 0)
        strcpy(name, "[unknown]");
        printf("%s  %s\n", addr, name);
    }

    free( ii );
    close( sock );
    return 0;
}
```

Passiamo a spiegare brevemente il programma appena visto. Per poter eseguire il programma su una distribuzione linux apriamo un terminale ci posizioniamo nel path in cui è presente il programma e digitiamo :

```
gcc -o scan scan.c -lbluetooth
./simplescan
```

Gli indirizzi dei dispositivi bluetooth vengono memorizzati all'interno di una struttura dati chiamata bdaddr_t

```
typedef struct {
        uint8_t b [6];
__attribute__} ((Imballato)) bdaddr_t;
```

BlueZ fornisce due utili funzioni per manipolare questi indirizzi e in particolare str2ba la quale trasforma una stringa della forma XX:XX:XX:XX:XX:XX in un indirizzo mentre ba2str contrario cioè trasforma un indirizzo in una stringa.
La prima cosa che deve essere fatta dal programma è quella di individuare un device locale che può essere utilizzato, questo compito è svolto dalla funzione hci_get_route la quale recupera il numero di risorse del primo adattatore Bluetooth disponibile.
Per essere effettuate le operazioni Bluetooth richiedono l'uso di un socket aperto. hci_open_dev è una funzione che apre un socket Bluetooth. Per essere chiari, il socket aperto da hci_open_dev rappresenta una connessione con il microcontroller sull'apposito adattatore locale, e non una connessione a un dispositivo Bluetooth remoto.
Dopo aver scelto l' interfaccia Bluetooth locale da utilizzare , il programma è pronto per la scansione dei dispositivi Bluetooth nelle vicinanze. Nell'esempio, hci_inquiry esegue una ricerca di e restituisce un elenco dei dispositivi rilevati e alcune informazioni di base su di loro.Queste informazioni saranno memorizzate in una struttura dati di nome inquiry_info .

```
typedef struct {
   bdaddr_t bdaddr;
   pscan_rep_mode uint8_t;
   pscan_period_mode uint8_t;
   pscan_mode uint8_t;
   uint8_t dev_class [3];
```

uint16_t clock_offset;

__attribute__} ((Imballato)) inquiry_info;

Una volta che un elenco dei dispositivi Bluetooth nelle vicinanze e il loro indirizzo è stato trovato, il programma determina i nomi associati con questi indirizzi e li presenta all'utente. La funzione hci_read_remote_name è usata per questo scopo.
Come abbiamo accennato prima dopo aver trovato un device con il quale poter comunicare nelle vicinanze il passo successivo per un programmatore è quello di iniziare la comunicazione vera e propria. Quindi ci si passerà alla scelta del protocollo da utilizzare per effettuare lo scambio di informazioni. I protocolli citati prima sono RFCOMM e L2CAP vediamo quindi 2 esempi nei quali vengono utilizzati questi 2 protocolli.
Vediamo prima il client e il server che usano il protocollo rfcomm e poi quelli che usano il protocollo L2CAP.

rfcomm-server.c

```
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/rfcomm.h>

int main(int argc, char **argv)
{
    struct sockaddr_rc loc_addr = { 0 }, rem_addr = { 0 };
    char buf[1024] = { 0 };
    int s, client, bytes_read;
    socklen_t opt = sizeof(rem_addr);

    // allocate socket
    s = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);

    // bind socket to port 1 of the first available
    // local bluetooth adapter
    loc_addr.rc_family = AF_BLUETOOTH;
    loc_addr.rc_bdaddr = *BDADDR_ANY;
    loc_addr.rc_channel = (uint8_t) 1;
    bind(s, (struct sockaddr *)&loc_addr, sizeof(loc_addr));

    // put socket into listening mode
    listen(s, 1);

    // accept one connection
    client = accept(s, (struct sockaddr *)&rem_addr, &opt);
```

```
    ba2str( &rem_addr.rc_bdaddr, buf );
    fprintf(stderr, "accepted connection from %s\n", buf);
    memset(buf, 0, sizeof(buf));

    // read data from the client
    bytes_read = read(client, buf, sizeof(buf));
    if( bytes_read > 0 ) {
        printf("received [%s]\n", buf);
    }

    // close connection
    close(client);
    close(s);
    return 0; }
```

rfcomm-client.c

```
#include <stdio.h>
#include <unistd.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/rfcomm.h>

int main(int argc, char **argv)
{
    struct sockaddr_rc addr = { 0 };
    int s, status;
    char dest[18] = "01:23:45:67:89:AB";

    // allocate a socket
    s = socket(AF_BLUETOOTH, SOCK_STREAM, BTPROTO_RFCOMM);

    // set the connection parameters (who to connect to)
    addr.rc_family = AF_BLUETOOTH;
    addr.rc_channel = (uint8_t) 1;
    str2ba( dest, &addr.rc_bdaddr );

    // connect to server
    status = connect(s, (struct sockaddr *)&addr, sizeof(addr));

    // send a message
    if( status == 0 ) {
        status = write(s, "hello!", 6);
    }

    if( status < 0 ) perror("uh oh");

    close(s);
    return 0;
}
```

Di seguito il client e il server che usano il protocollo L2CAP :

l2cap-server.c

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/l2cap.h>

int main(int argc, char **argv)
{
    struct sockaddr_l2 loc_addr = { 0 }, rem_addr = { 0 };
    char buf[1024] = { 0 };
    int s, client, bytes_read;
    socklen_t opt = sizeof(rem_addr);

    // allocate socket
    s = socket(AF_BLUETOOTH, SOCK_SEQPACKET, BTPROTO_L2CAP);

    // bind socket to port 0x1001 of the first available
    // bluetooth adapter
    loc_addr.l2_family = AF_BLUETOOTH;
    loc_addr.l2_bdaddr = *BDADDR_ANY;
    loc_addr.l2_psm = htobs(0x1001);

    bind(s, (struct sockaddr *)&loc_addr, sizeof(loc_addr));

    // put socket into listening mode
    listen(s, 1);

    // accept one connection
    client = accept(s, (struct sockaddr *)&rem_addr, &opt);

    ba2str( &rem_addr.l2_bdaddr, buf );
    fprintf(stderr, "accepted connection from %s\n", buf);

    memset(buf, 0, sizeof(buf));

    // read data from the client
    bytes_read = read(client, buf, sizeof(buf));
    if( bytes_read > 0 ) {
        printf("received [%s]\n", buf);
    }

    // close connection
    close(client);
    close(s);
}
```

l2cap-client.c

```
#include <stdio.h>
#include <string.h>
#include <sys/socket.h>
#include <bluetooth/bluetooth.h>
#include <bluetooth/l2cap.h>

int main(int argc, char **argv)
{
    struct sockaddr_l2 addr = { 0 };
    int s, status;
    char *message = "hello!";
    char dest[18] = "01:23:45:67:89:AB";

    if(argc < 2)
    {
        fprintf(stderr, "usage: %s <bt_addr>\n", argv[0]);
        exit(2);
    }

    strncpy(dest, argv[1], 18);

    // allocate a socket
    s = socket(AF_BLUETOOTH, SOCK_SEQPACKET, BTPROTO_L2CAP);

    // set the connection parameters (who to connect to)
    addr.l2_family = AF_BLUETOOTH;
    addr.l2_psm = htobs(0x1001);
    str2ba( dest, &addr.l2_bdaddr );

    // connect to server
    status = connect(s, (struct sockaddr *)&addr, sizeof(addr));

    // send a message
    if( status == 0 ) {
        status = write(s, "hello!", 6);
    }

    if( status < 0 ) perror("uh oh");

    close(s);
}
```

I programmi che abbiamo appena visto hanno lo stesso scopo quello di mostrare una semplice trasmissione di dati utilizzando 2 protocolli differenti RFCOMM e L2CAP . Ovviamente perché tutto funzioni alla perfezione deve essere mandato in esecuzione prima il server il quale dopo aver aperto un soket rimane in ascolto su di esso, a questo punto può partire il client il quale invia una richiesta di connessione e invia una stringa di dati (in questo caso “hello”) al server che si risveglia dallo stato lissen accetta la connessione e inizia a leggere i dati ricevuti stampando il tutto a video. Per scenari di utilizzo semplici, le uniche differenze sono il tipo di socket specificato, la famiglia di protocolli, e la struttura di indirizzamento. Per impostazione predefinita, le connessioni L2CAP forniscono collegamenti datagram-oriented affidabili e con i pacchetti consegnati in ordine, dove il tipo di socket è SOCK_SEQPACKET, e il protocollo è BTPROTO_L2CAP. La struttura d'indirizzamento struct sockaddr_l2 differisce leggermente dalla struttura di indirizzamento RFCOMM.

6.3 BNEP

Nel capitolo 4 abbiamo parlato dello stack bluetooth descrivendo brevemente i protocolli che lo costituiscono. Come per il modello ISO/OSI anche lo stack bluetooth è costituito da layer , che racchiudono uno o più aspetti fra loro correlati come per esempio trovare dispositivi vicini, scoprire quali servizi offrono ecc. Tra i protocolli visti fino ad ora quello che più si presta al nostro scopo è L2CAP.

Questo protocollo però è un protocollo al livello data link serve quindi un protocollo comune che sia in grado di incapsulare i pacchetti provenienti dal Network layer e trasportarli direttamente sul data link layer Bluetooth rappresentato da L2CAP.
Il protocollo in questione è il Bluetooth Network Encapsulation Protocol (BNEP).

Nella figura che segue possiamo vedere lo stack bluetooth con l' inserimento del protocollo Bnep, al di sopra del quale troviamo il protocollo IP andando ancora sopra il TCP/UDP fino ad arrivare a una qualsiasi Network Application che sfrutta questi protocolli.

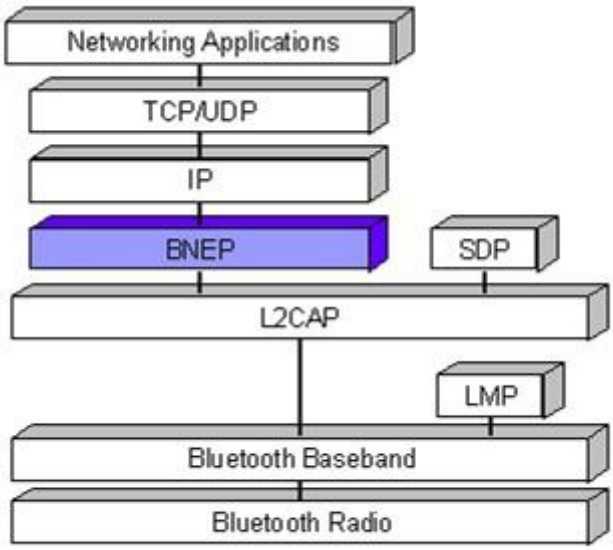


Figura numero 9 stack con bnep

Bnep incapsula i pacchetti di protocolli di rete diversi, che vengono trasportati direttamente sopra L2CAP , dove L2CAP rappresenta il livello data link .

Mette a disposizione Il supporto per protocolli di rete comuni, come IPv4, IPv6 ed altri protocolli di rete esistenti o emergenti. L'uso di BNEP per il trasporto di un pacchetto Ethernet è illustrato in questa figura .

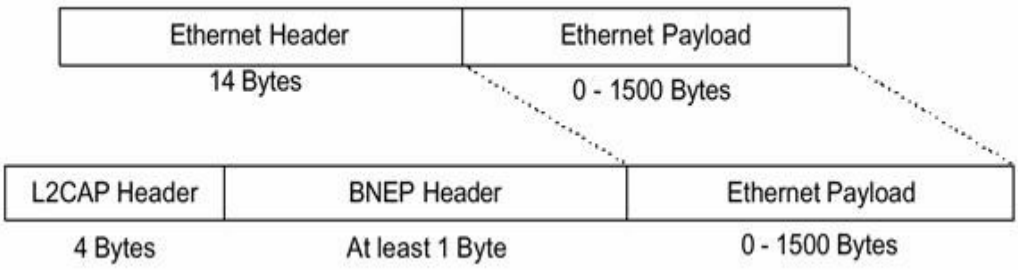
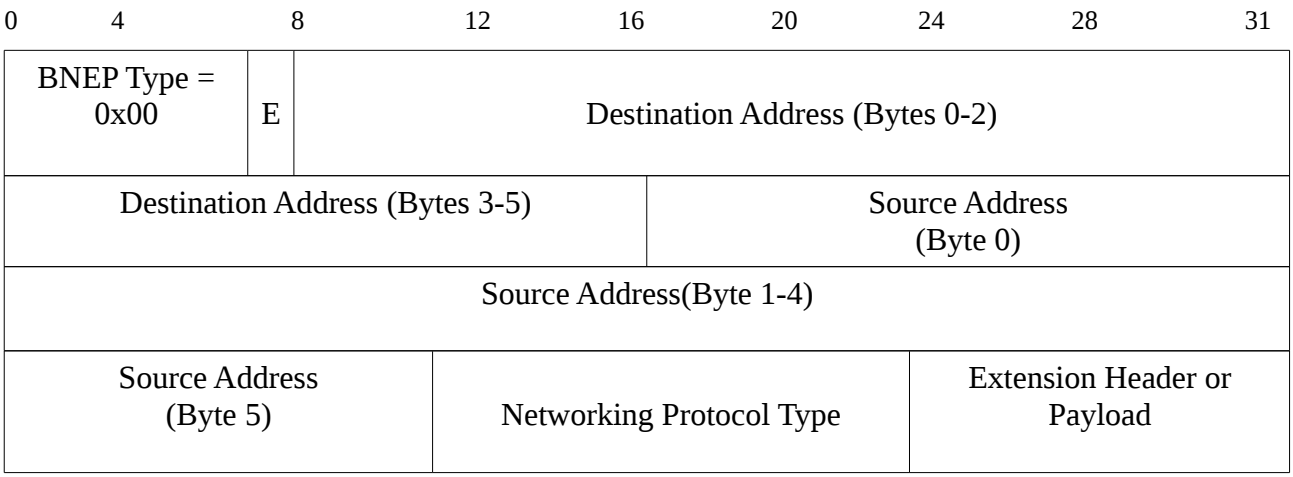


Figura numro 10 pacchetto bnep

BNEP elimina e sostituisce l'intestazione Ethernet con la propria intestazione. Il payload Ethernet rimane invariato. Infine, sia l'intestazione BNEP che il payload Ethernet viene incapsulato da L2CAP e viene inviato sui mezzi di comunicazione Bluetooth. La dimensione massima che può assumere il payload che BNEP accetterà dallo strato superiore è pari a 1691 byte, meno 191 byte

riservato per le intestazioni BNEP.
Questa dimensione è dettata dallo strato inferiore L2CAP il cui minimo Maximum Trasmission Unit (MTU) è proprio 1691 byte In questo modo si può essere certi che c' è sufficiente spazio per trasmettere tutti i pacchetti BNEP.
Questo protocollo offre funzionalità che sono simili alle funzionalità fornite da Ethernet (EthernetII / DIX Framing / IEEE 802.3).
Nella tabella sottostante possiamo vedere il BNEP_GENERAL_ETHERNET packet, questi pacchetti vengono utilizzati per trasportare i pacchetti Ethernet da e per le reti Bluetooth.



Le tabelle successive descrivono brevemente i campi presenti nella tabella che abbiamo già visto, prendendo in esame il valore che questi campi possono assumere e il compito che svolgono.

Bnep Type:

Valore	Descrizione
0x00	BNEP Type sono 7 bit che identificano il tipo di valore header contenuto in questo pacchetto. Deve esere settato come BNEP_GENERAL_ETHERNET

Extension Flag (E):

Valore	Descrizione
0x0 - 0x1	Un bit che identifica se uno o più extension header seguono il bnep header prima del playload. Se questo flag è settato a 0x1 uno o più extension header seguono il bnep header altrimenti se è settato a 0x0 il bnep header è seguito solo ed esclusivamente dal payload

Destination Address:

Valore	Descrizione
0xXXXXXXXXXXXX	Sono 48 bit che identificano l'indirizzo IEEE di destinazione del pacchetto BNEP e sono contenuti nel payload.

Source Address:

Valore	Descrizione
0xXXXXXXXXXXXX	Sono 48 bit che identificano l'indirizzo IEEE del mittente del pacchetto BNEP e sono contenuti nel payload

Networking Protocol Type:

Valore	Descrizione
0xXXXX	Sono 16 bit che identificano il tipo di protocollo network contenuto nel payload.

6.4 Pand daemon

Lo strumento più importante che bluez ci mette a disposizione è il demone pand questo demone permette di collegare il computer alla rete ethernet usando il device bluetooth qui di seguito ho riportato la pagina di manuale di questo demone.
Come possiamo vedere sono molte le opzioni che si possono utilizzare :

NAME

pand - BlueZ Bluetooth PAN daemon

DESCRIPTION

The pand PAN daemon allows your computer to connect to ethernet networks using Bluetooth.

SYNPOSIS

pand <options>

OPTIONS

--show --list -l

Show active PAN connections

--listen -s

Listen for PAN connections

--connect -c <bdaddr>

Create PAN connection

--search -Q[duration]

Search and connect

--kill -k <bdaddr>

Kill PAN connection

--killall -K

Kill all PAN connections

--role -r <role>

Local PAN role (PANU, NAP, GN)

--service -d <role>

Remote PAN service (PANU, NAP, GN)

--ethernet -e <name>

Network interface name

--device -i <bdaddr>

Source bdaddr

--nosdp -D

Disable SDP

--encrypt -E

Enable encryption

--secure -S

Secure connection

--master -M

Become the master of a piconet

--nodetach -n

Do not become a daemon

--persist -p[interval]

Persist mode

--cache -C[valid]

Cache addresses

--pidfile -P <pidfile>

Create PID file

--devup -u <script>

Script to run when interface comes up

--devdown -o <script>

Script to run when interface comes down

--autozap -z

Disconnect automatically on exit

SCRIPTS

The devup/devdown script will be called with bluetooth device as first argument and bluetooth destination address as second argument.

Il demone pand quindi permette di realizzare una PAN(Personal Area Network) nella quale i dispositivi partecipanti possono avere ruoli differenti. Un dispositivo può partecipare alla rete con ruolo di :

- Pan User PANU è quel terminale che prende parte alla piconet con il ruolo di slave ed è il device che sfrutta i servizi degli altri terminali con ruoli di GN o NAP

- Group Ad-hoc Network (GN) è quel terminale che partecipa alla piconet con ruolo di master e permette lo scambio di informazioni tra più PANU in modo simile ad una rete peer-to-peer. 7 è il numero massimo di terminali bluetooth che partecipano.

- Network Access Point (NAP) partecipa alla piconet come master e mette a disposizione un proxy un router o un bridge tra una rete esterna di solito una rete LAN e 7 device PANU

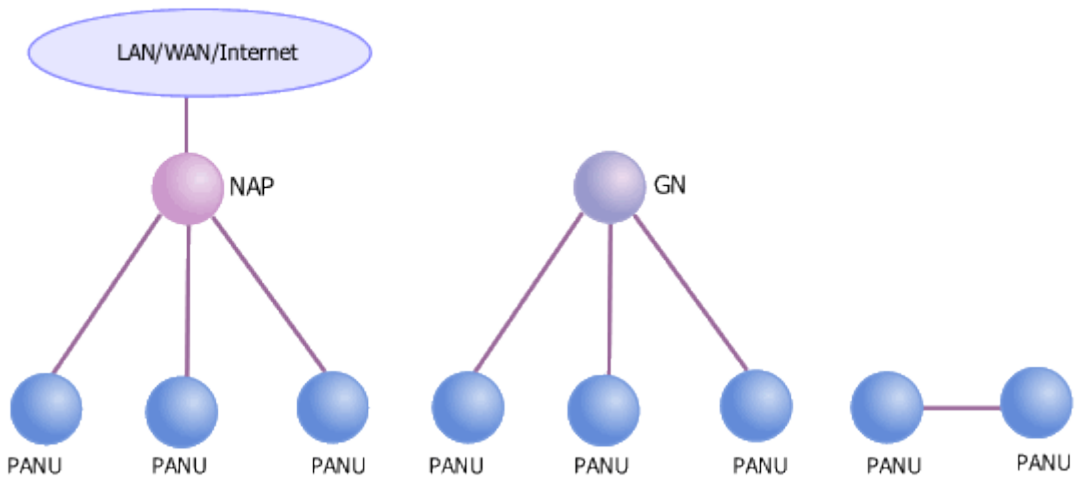


figura numero

Capitolo 7

7.1 Manuale D' uso

La tesi è stata sviluppata utilizzando 2 computer, sui quali è stato installato un sistema operativo linux Ubuntu. Per poter usufruire degli strumenti descritti nel capitolo precedente si possono sceglire 2 vie, utilizzando i repository messi a disposizione dal sistema operativo dove poter scaricare tutti i programmi necessari al corretto funzionamento del sistema oppure scaricando sul sito del progetto bluez il tar.gz che contiene tutti gli strumenti e mette a disposizione le librerie per lo sviluppo.
Nel pacchetto ovviamente ci sono i file README e INSTALL che spiegano in maniera dettagliata i passi necessari alla configurazione del sistema.
Nel caso in cui si volessero utilizzare i repository i pacchetti da scaricare sono i seguenti:

- bluez-compat : mette a disposizione il demone pand descritto precedentemente
- bluez-utils : mette a disposizione tool quali hcitool e hciconfig rispettivamente per la scansione di dispositivi presenti nelle vicinanze e per la configurazione del device bluetooth locale
- bluez-hcidump : utilizzato per debag della rete

per installare questi pacchetti nel sistema ubuntu aprire una finestra di terminale e digitare :

sudo apt-get install bluez-compat bluez-utils bluez-hcidump

se tutto è andato a buon fine digitando il comando hciconfig ci dovrebbero apparire a video informazioni sul device locale come nel mio caso.

```
root@caos:~/ hciconfig
hci0:   Type: BR/EDR  Bus: USB
        BD Address: 00:1E:52:DC:F7:2A  ACL MTU: 384:8  SCO MTU: 64:8
        UP RUNNING PSCAN
        RX bytes:1930 acl:0 sco:0 events:55 errors:0
        TX bytes:728 acl:0 sco:0 commands:55 errors:0
```

mentre digitando il comando hcitool scan verrà stampato a video la lista dei dispositivi presenti nelle vicinanze come nel mio caso :

root@caos:~ / hcitool scan
Scanning ...
	00:22:43:F3:E9:D7	franki-0

per poter procedere con questa guida si deve eseguire il pairing tra i dispositivi bluetooth utilizzati . Aprire il terminale su tutti e 2 i computer uno di loro sarà utilizzato come PANU mentre l' altro come NAP.
Digitare su tutti e 2 sudo -s ci sarà una richiesta di password questo servirà a diventare amministratore della macchina .
Ovviamente il dispositivo utilizzato come server e quindi NAP sarà collegato alla rete esterna mentre il dispositivo client PANU avrà solo ed esclusivamente l' interfaccia bluetooth abilitata per verificare l' effettivo funzionamento .

Server :
pand --listen --master --role NAP
viene messo il server in ascolto con ruolo di NAP

Client :

hcitool scan
Scanning ...
	00:1E:52:DC:F7:2A	caos-0

pand –connect 00:1E:52:DC:F7:2A

dopo la scansione il client si collega al device bluetooth presente con pand – connect in questo modo viene mandato in esecuzione il modulo bnep su entrambi i divice. Si procede quindi alla configurazione dell'interfaccia virtuale Bnep su entrambi i device .

Server :
ifconfig bnep0 192.168.1.30 netmask 255.255.255.0

Client :
ifconfig bnep0 192.168.1.30 netmask 255.255.255.0
route add defaul gw 192.168.1.30

con il comando route si può vedere la situazione che si è venuta a creare
root@caos:~/route
Tabella di routing IP del kernel

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.1.0	*	255.255.255.0	U	0	0	0	bnep0
default	*	caos.local	UG	1	0	0	bnep0

L' unico tassello mancante è l' abilitazione del' ip forwording lancio quindi sul server uno script che lo abilita.

```
#!/bin/sh

echo "1" > /proc/sys/net/ipv4/ip_forward
for f in /proc/sys/net/ipv4/conf/*/rp_filter ; do
    echo 1 >> $f
    #echo $f
done

iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE -v
iptables -A FORWARD -i bnep0 -j ACCEPT -v
iptables -A FORWARD -m state --state ESTABLISHED,RELATED -j ACCEPT -v
```

Quest'ultima operazione permette al client di connettersi alla rete esterna sfruttando il server come gateway.
Un ultimo accorgimento editare il file /etc/resolv.conf scrivendo nameserver indirizzo
dove indirizzo è il nome del nameserver.

Capitolo 8

Conclusioni

L' obiettivo che ci siamo prefissi con questa tesi è quello di utilizzare un interfaccia bluetooth in modo da poter creare una piconet dove il dispositivo master è collegato ad una LAN e funge da NAP(Network Access Point) per i dispositivi slave che si collegano .

In questo modo un qualsiasi terminale sul quale è presente un interfaccia bluetooth è capace di limitare i consumi energetici continuando ad offrire una discrete velocità di connessione.

Come dispositivi hardware sono stati utilizzati due laptop computer con architettura Intel x86 sui quali ho installato una distribuzione Linux Ubuntu 11.10 .

Solo uno dei 2 computer è stato collegato alla rete mentre sul' altro ho disabilitato tutte le interfacce per focalizzarmi solo su quella bluetooth e verificarne l' effettivo funzionamento.

La gestione dell'interfaccia bluetooth da parte di questa distribuzione è stata realizzata per mezzo di Bluez un software libero rilasciato anche esso sotto licenza GPL.
Con l'ausilio di questi strumenti posso affermare che l' obiettivo è stato raggiunto e che è quindi possibile rimanere connessi utilizzando un interfaccia bluetooth.
Anche se la velocità di connessione è di gran lunga inferiore a quella messa a disposizione dalle altre interfacce di rete ha il pregio di essere molto economica e di mantenere i consumi energetici bassi, questo pregio si sposa a meraviglia con il modello ABPS il quale come abbiamo già spiegato si colloca in uno scenario di mobilità urbana dove il risparmio energetico ha sicuramente la sua importanza .