Alma Mater Studiorum · Università di Bologna

FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI
Corso di Laurea Magistrale in Informatica

HIDDEN-PROXY PER COMUNICAZIONI SEAMLESS DA DISPOSITIVI MULTI-HOMED

Tesi di Laurea in Architettura degli elaboratori

Relatore:
Chiar.mo Prof.
Vittorio Ghini

Presentata da:
Giulio Grassi

Sessione III
Anno Accademico 2010-2011

Parole chiave:
Wireless, mobilitá, multi-homing, protocolli, trasparenza

Introduzione

Negli ultimi anni si é assistito ad una rapida diffusione di device che offrono agli utenti gli stessi servizi di un PC ma che fanno della possibilitá di spostarsi in libertá uno dei loro punti di forza. Netbook, tablet, smartphone sono entrati nella vita quotidiana di ogni persona accompagnandola in ogni azione e in ogni luogo. Parallelamente, la diffusione di una miriade di servizi su Internet ha creato una vera e propria dipendenza delle persone alla rete, dipendenza che le porta a voler rimanere connesse non solo all'interno della propria casa o del proprio ufficio, ma in ogni posto in cui esse si trovino. Vogliono poter connettere il loro device mobile ad Internet ovunque esse siano e in ogni istante, in modo da non perdersi l'ultima news online, il film che stanno guardando in streaming, l'ultimo aggiornamento su Facebook di un amico o la videochiamata con la fidanzata.
Purtroppo peró non esiste ad oggi una tecnologia in grado di poter garantire questo tipo di connessione a tutte le persone; alcune tecnologie infatti garantiscono elevate prestazioni in termini di banda ma comportano una bassa copertura che quindi non puó sostenere la mobilitá degli utenti, mentre altre tecnologie hanno puntato ad un raggio d'azione maggiore, in modo da offrire un buon servizio di supporto alla mobilitá, andando peró a perdere in prestazioni.
Non potendo avere un'unica tecnologia che ci possa garantire supporto alla mobilitá assieme ad elevate prestazioni, la soluzione migliore sembra ad oggi essere quella di utilizzare assieme tutte queste tecnologie, in modo da poter mediare i limiti di una tecnologia con i benefici dell'altra e viceversa, cosí

da potersi muovere in libertá e rimanere connessi ad Internet mantenendo un'elevata qualitá delle trasmissioni. Il problema che peró nasce con un approccio del genere consiste nel fatto che senza un'architettura di supporto non é possibile utilizzare assieme piú interfacce di rete di uno stesso dispositivo.
HPforMSC (Hidden Proxy for Multihomed Seamless Communication), il sistema oggetto di questa tesi, é appunto un'architettura per il supporto alla mobilitá che permette l'utilizzo contemporaneo di differenti tecnologie wireless da parte dello stesso dispositivo, in modo da garantire la piú ampia copertura possibile, mantenendo prestazioni ottimali.

La presentazione di questo lavoro verrá suddivisa nelle seguenti parti:

- Una descrizione dello scenario in cui HPforMSC si trova ad operare: le tecnologie wireless piú in voga, i nuovi requisiti legati al concetto di mobilitá degli utenti.
- Una panoramica sulle soluzioni adottate ad oggi per il supporto alla mobilitá, analizzando le prestazioni, i pregi e i difetti.
- Una presentazione dei problemi che HPforMSC cerca di risolvere.
- Una carrellata degli strumenti utilizzati per la realizzazione di HPforMSC.
- Una descrizione approfondita dell'architettura di HPforMSC, con un'analisi di ogni sua parte e dei meccanismi interni che lo governano.
- Alcuni dettagli sulle parti piú salienti ed interessanti dell'implementazione di HPforMSC.
- Una valutazione di HPforMSC comprendente alcuni test effettuati e alcune considerazioni sulle prestazioni.
- Alcuni cenni sui possibili sviluppi futuri e migliorie attuabili.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Scenario

1.1 Tecnologie Wireless

Una rete wireless permette alle persone di comunicare e accedere ad una moltitudine di informazioni senza la necessitá di utilizzare cavi, liberandole quindi dalla costrizione di dover essere fisicamente attaccati al punto di accesso della rete. Le reti wireless liberano le persone dal vincolo di staticitá che invece caratterizza le reti su cavo. I dispositivi attraverso i quali una persona puó accedere ad una rete wireless sono innumerevoli, partendo dai classici Personal Computers e laptops ai piú recenti smartphone, tablet, PDAs (Personal Device Assistants), per non parlare di tutti quei device che offrono un servizio specifico tramite reti wireless, come stampanti, fax, scanner ...

1.1.1 Tipologia di reti wireless

Il mondo delle tecnologie wireless é molto ampio e variegato e negli ultimi anni nuove tipologie di reti si sono o si stanno diffondendo rapidamente. Appunto per la vastitá di questo mondo, prima di iniziar a parlare di alcune di queste tecnologie, é forse meglio fare un po' di ordine. Le reti wireless possono essere molto differenti tra di loro, in base a fattori come bandwidth, consumo energetico, device tipici che vi possono accedere, numero di device massimo, raggio d'azione. Proprio quest'ultima caratteristica é la discriminante prin-

cipale nella categorizzazione delle reti wireless forse piú comune; infatti e' possibile suddividere il mondo delle reti wireless nelle seguenti categorie:

- *WPANS* (Wireless Personal Area Networks): queste reti vengono dette personali perché hanno lo scopo di collegare tra di loro device in possesso dell'utente o che si trovano in prossimitá (una decina di metri) dalla persona stessa. Le tecnologie piú comuni nell'ambito delle WPANS sono infrarossi e Bluetooth (IEEE802.15).

- *WLANS* (Wireless Local Area Networks): reti create tipicamente per offrire connettivitá verso Internet o per connettere tra di loro device situati nella stessa area, per esempio all'interno della stessa abitazione, lo stesso ufficio... In questo settore la tecnologia piú conosciuta é sicuramente il WiFi (IEEE802.11).

- *WMANS* (Wireless Metropolitan Area Networks): come dice il nome, questa tipologia di rete ha un raggio d'azione molto piú ampio, che puó arrivare a raggiungere un'intera cittadina. In questo caso la tecnologia predominante é WIMAX (IEEE802.16).

- *WWANS* (Wireless Wide Area Networks): quest'ultima categoria comprende tutte quelle reti, tipicamente satellitari e cellulari, il cui raggio d'azione supera l'area urbana e puó raggiungere anche intere regioni o, nel caso di tecnologie satellitari, aree anche di piú ampia estensione. In questa categoria rientrano tutte le tecnologie definite 2G (GSM), 3G (UMTS, HSPA) e 4G.

Come giá accennato, il mondo delle tecnologie wireless é molto ampio. Prendendo in considerazione quelle che forniscono connettivitá ad Internet a device in un raggio d'azione medio - ampio (WLAN, WMAN e WWAN), la maggior parte di device in commercio attualmente come PC, tablet, smartphone supportano una ristretta gamma di reti wireless, concentrandosi solitamente sulle tecnologie riportate qui di seguito:

- *WiFi*: é forse la tecnologia wireless piú conosciuta e diffusa al mondo, basti pensare che ben il 24% del traffico dati negli Stati Uniti passa per WiFi[1]. Questa tecnologia, che si basa sullo standard IEEE802.11, permette a tutti i device connessi in una rete locale del raggio di qualche decina di metri di avere accesso ad Internet tipicamente tramite un access point, letteralmente punto di accesso della rete locale ad Internet. Oltre a questa modalitá, detta "managed", WiFi permette anche un'altra modalitá, l'Ad-Hoc, tipicamente meno diffusa della precedente e dedicata alle reti temporanee costituite da pochi nodi; questa modalitá non prevede un access point che gestisce la rete, che invece viene gestita dagli stessi nodi connessi. La grande peculiaritá delle reti ad-hoc é il fatto di non richiedere appunto un'infrastruttura coordinatrice, il che quindi le rende meno costose e piú facilmente sviluppabili. Nonostante ció, di norma questa modalitá non viene utilizzata molto frequentemente, perché vi sono ancora dei problemi non risolti completamente, soprattutto nella fase di routing, che la rendono meno efficiente della modalitá managed. Negli ultimi anni le reti ad-hoc sono state comunque soggette a numerevoli studi, soprattutto nel campo veicolare, le cosiddette reti Vanet[2].

- *WiMax*(Worldwide Interoperability for Microwave Access) : é una tecnologia basata sullo standard IEEE802.16, nata attorno al 2001, quindi piú recente del WiFi, la quale, rispetto a quest'ultima, permette un range piú ampio, dell'ordine dei 3-10 chilometri [3] e un bandwidth massimo pari a 70-75 Mbps. WiMax é una tecnologia che si sta rapidamente espandendo, grazie anche agli investimenti fatti negli ultimi anni dai vari provider di rete, i quali stanno costantemente aumentando la copertura delle reti WiMax.

[1]Fonte: AdMob Mobile Metrics, May 2010

[2]Veichular Ad-Hoc Network

[3]Il range trasmissivo cresce fino a 30-50 chilometri quando i due nodi sono in Line-of-Sight (LOS)

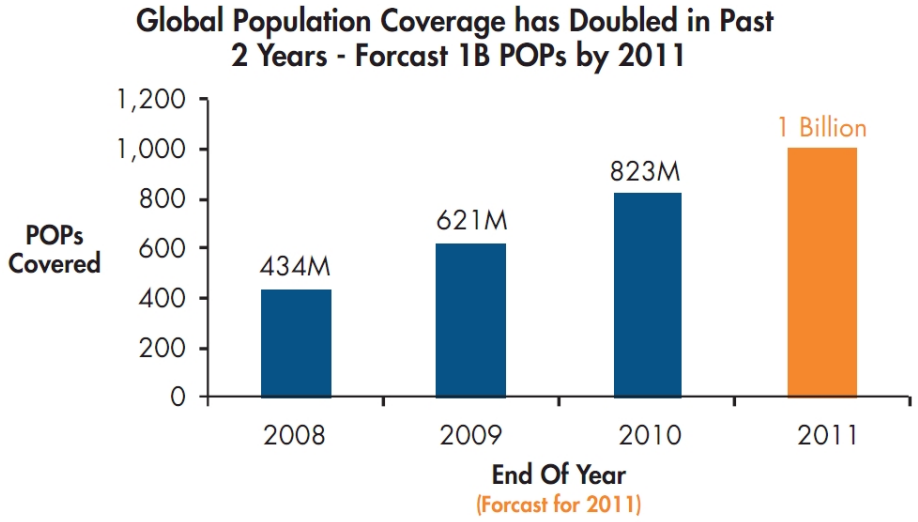

Figura 1.1: Numero persone che vivono in aree con copertura WiMax. Fonte: WiMAX Forum®Industry Research Report, May 2011

- *UMTS, HSPA*: Tecnologie di telefonia mobile dette di terza generazione (3G), le quali permettono l'accesso ad Internet agli utenti presenti in un'area molto piú ampia rispetto alle tecnologie precedenti, ma con prestazioni in termini di bandwidth solitamente inferiori. Attualmente sono tecnologie con un bacino di utenza molto ampio[4] e molto utilizzate soprattutto nei dispositivi mobili quali smartphone, tablet ... , grazie proprio al fatto che il raggio d'azione risulta essere molto ampio. Periferiche esterne che supportano queste tipologie di connessione, non direttamente integrate nei dispositivi precedenti, hanno avuto una grande diffusione anche in device quali PC e notebook, grazie all'utilizzo delle cosiddette "chiavette USB 3G", periferiche USB che appunto forniscono connettivitá di tipo UMTS o HSPA.

- *LTE e MobileWimax 2 (WirelessMAN-Advanced)*: Sono tecnologie molto recenti le quali hanno da poco ottenuto la qualifica di tecnologie cellu-

[4]Si stima che il 45% della popolazione mondiale viva in un'area con copertura 3G. Fonte: ITU World Telecommunication/ICT Indicators database

lari di quarta generazione (4G) [5]. Nate come evoluzione rispettivamente di UMTS e WiMax, sono caratterizzate da supportare la mobilitá degli utenti in ampie aree, mantenendo molto elevata la capacitá massima di bandwidth[6]. Sono tecnologie in ampia diffusione, che stanno attirando l'attenzione delle maggiori compagnie di telecomunicazioni, le quali nel 2011 hanno investito solamente in LTE ben 3 miliardi di dollari. Basti pensare che le previsioni di crescita degli utenti MobileWimax 2 si aggirano attorno al 51,1% (CAGR[7]) e quelle dei ricavi per entrambe le tecnologie si aggirano attorno al 299% (CAGR)[8].

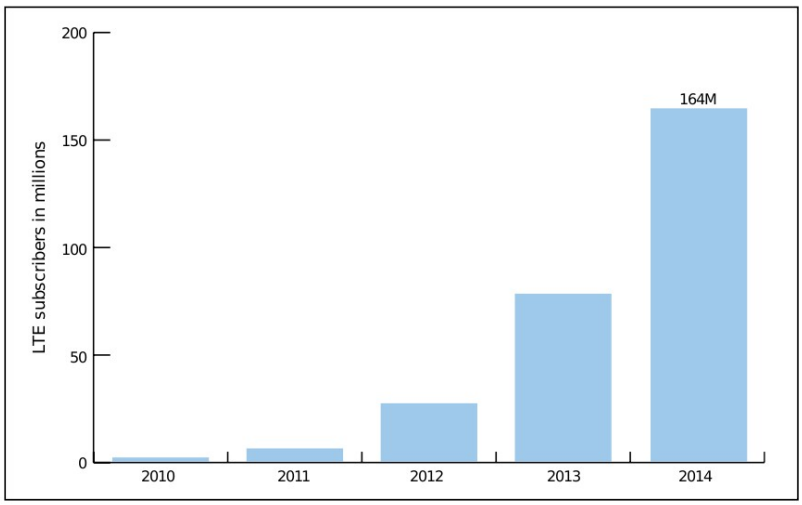

Figura 1.2: Previsione di crescita degli utenti della tecnologia LTE. Fonte: Infonetics Research, LTE Infrastructure and Subscribers - Biannual Worldwide and Regional Market Size and Forecasts, October 2010

[5]ITU World Radiocommunication Seminar highlights future communication technologies, Dicembre 2010

[6]Secondo requisiti UTI per 4G, con alta mobilitá si dovrebbero ottenere all'incirca 100Mbit/s, mentre con utenti pressoché statici si dovrebbero raggiungere picchi di 1 Gbit/s

[7]Compound Annual Growth Rate

[8]Fonte: Renub Research report on "4G (LTE and WiMAX) Service Revenue/Market Analysis and its Opportunities for Industries"

1.2 Mobilitá e nuovi device

Una rete wireless elimina la staticitá delle persone che vi sono collegate. Mentre in passato, quando si utilizzava esclusivamente WiFi su dispositivi come notebook, questo principalmente significava potersi spostare di qualche metro, per esempio da una scrivania all'altra, o da una stanza all'altra della propria casa, negli ultimi anni le cose sono notevolmente cambiate. Da un lato, sono nate le nuove tecnologie descritte in precedenza, caratterizzate da un raggio d'azione maggiore, dall'altro abbiamo assistito alla rapida diffusione di device simili a PC ma che fanno della possibilitá di spostarsi in libertá uno dei loro punti di forza. Si é iniziato con l'evoluzione dei dispositivi mobili per eccellenza, i cellulari, in device piú "intelligenti", piú vicini a PC per capacitá di calcolo, i cosiddetti smartphone. Successivamente, nel settore dei device mobili, sono comparsi nuovi attori, i tablet, tra cui senza alcun dubbio il principale e piú famoso é l'IPad, della Apple. Tutti questi dispositivi, assieme ai comunque sempre presenti notebook e soprattutto netbook, hanno invaso la vita delle persone, offrendo loro servizi classici dei PC, ovunque essi siano, in macchina, in treno, in ufficio, all'aperto assieme agli amici. Facendo della mobilitá e della possibilitá di utilizzo in ogni luogo il loro punto di forza, tutti questi device mobili sono diventati onnipresenti nella vita delle persone, accompagnandole spesso in ogni fase della giornata.

Figura 1.3: Alcuni device mobili: smartphone, tablet, netbook

1.2.1 Risorse richieste dalle applicazioni

Alla contemporanea irruzione nel mercato di nuovi device mobili, negli ultimi anni si é assistito anche ad un aumento nella richiesta di bandwidth da parte degli utenti. Questo é avvenuto principalmente per due motivi: il primo consiste nel fatto che il numero di persone che vivono in aree con una qualsivoglia copertura per la connessione ad Internet é aumentata, grazie anche all'introduzione delle nuove tecnologie wireless ad ampio raggio come quelle di terza generazione. Il secondo motivo é invece da ricercare nel fat-

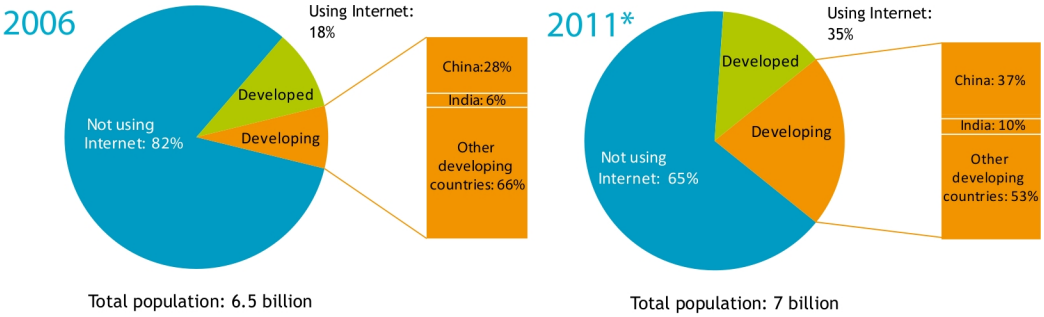

Figura 1.4: Utenti di Internet negli ultimi anni [9]

to che gli stessi utenti richiedono una banda maggiore rispetto al passato, a causa della diffusione di servizi ad alto consumo di bandwidth, come file sharing, P2P, streaming video come Youtube, Megavideo ... Gli stessi contenuti presenti su Internet richiedono maggior consumo di banda a causa del costante aumento delle loro dimensioni. Si é passati da pagine statiche contenenti principalmente testo o immagini, a pagine contenenti video, a volte interi film, che appunto causano un aumento del flusso di bytes che attraversa Internet.

Sempre connessi

Oltre all'aumento della richiesta di banda, un altro fattore é cambiato nella modalitá di accesso ad Internet. Infatti da una parte c'é stato un aumento delle applicazioni che richiedono una connessione ad Internet, basti pensare

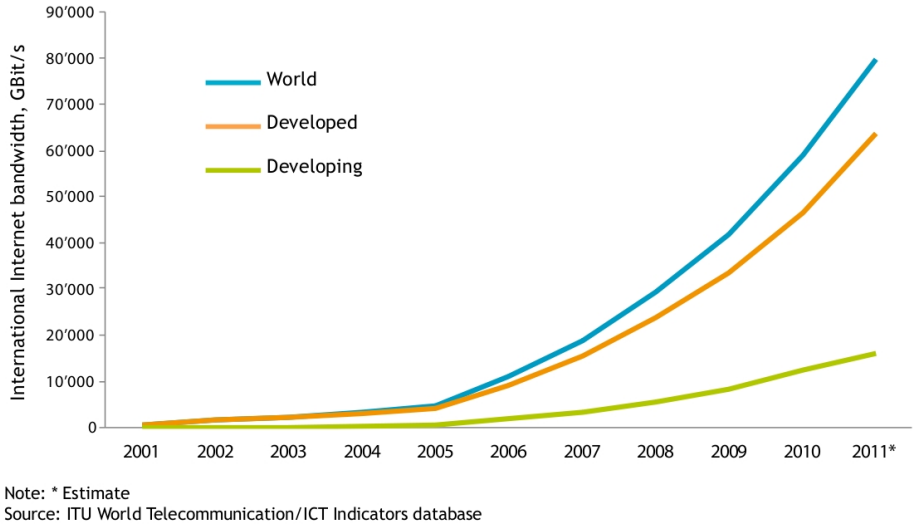

Figura 1.5: Bandwidth di Internet negli ultimi anni. Fonte: ITU World Telecommunication/ICT Indicators database. 2011

a tutti quei servizi che spostano parte delle funzionalitá di un filesystem dal device dell'utente a server presenti nella rete, come Dropbox, Google Documents, ICloud ... Dall'altra parte, mente in passato chi usufruiva di Internet solitamente si trovava comodamente seduto in ufficio o a casa, ora l'introduzione del concetto di mobilitá avvenuta con la diffusione di device mobili sta portando le persone a richiedere connettivitá ovunque esse si trovino, indipendentemente dai loro spostamenti. Le persone vogliono essere connesse in ogni momento e in ogni luogo esse si trovino. Chiedono di poter usufruire sempre di tutti i servizi a loro disposizione nel proprio device, sia che si trovino seduti davanti la scrivania di casa, sia che si trovino in macchina in mezzo al traffico, in un parco a fare jogging o al bar ...

Qualitá del canale

Oltre alla quantitá di banda e alla frequenza con cui si accede ad Internet, con la diffusione di servizi quali VoIP, streaming di film o di musica e tutti gli altri servizi ad alta interattivitá con l'utente tipici del web 2.0, hanno aumentato la loro importanza anche altri requisiti, collegati con la qualitá

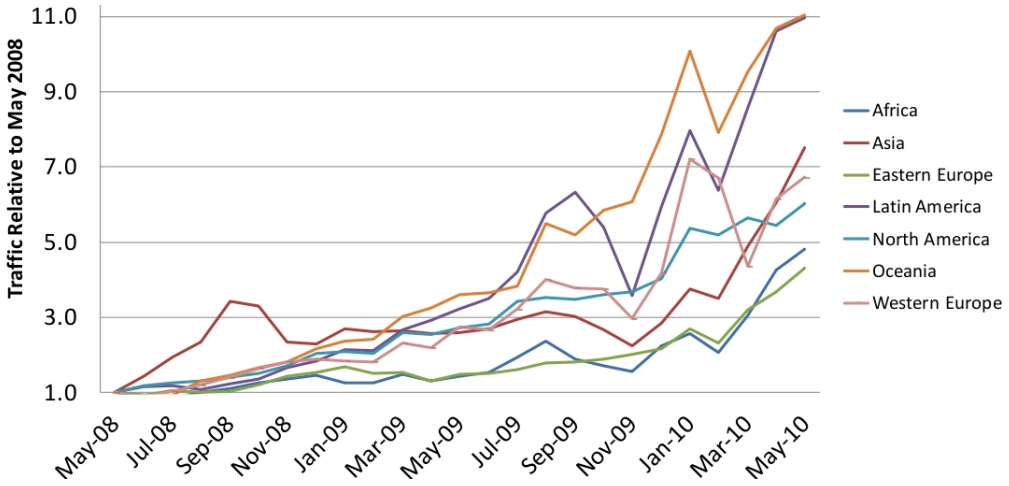

Figura 1.6: Crescita del traffico dati da device mobili dal 2008 a Maggio 2010. Fonte: AdMob Mobile Metrics, May 2010

del canale, come:

- *Latenza end-to-end*: consiste nel tempo necessario per trasmettere i dati tra i due estremi della comunicazione. Occorre che questo ritardo non sia eccessivo per la trasmissione dei dati, soprattutto per le applicazioni real-time; pensando per esempio ad un'applicazione per VoIP come Skype, nel caso la latenza sia eccessiva, la comunicazione risulterebbe incomprensibile perché i 2 interlocutori avrebbero difficoltá perfino ad individuare il momento esatto in cui l'altro sta parlando e quindi in cui occorre rimanere in silenzio, per non sovrapporre la propria voce e rendere appunto il dialogo pressoché impossibile.

- *Continuitá della connettivitá*: occorre che il servizio di connettivitá sia continuo. In questo caso, si possono individuare due requisiti separati ma di eguale importanza, l'intervallo di indisponibilitá e l'intervallo di continuitá. Il primo indica il tempo in cui una comunicazione si interrompe temporaneamente, mentre il secondo indica il tempo medio durante il quale la comunicazione non ha interruzioni. Mentre l'intervallo di indisponibilitá deve essere il piú piccolo possibile, l'intervallo di

continuitá deve essere massimizzato. Pensando sempre all'applicazione VoIP infatti, una trasmissione con picchi di bandwith molto elevati seguiti a frequenti intervalli con banda vicina allo zero o a lunghi periodi di silenzio (intervallo di indisponibilitá) risulta essere peggiore di una connessione con una banda mediocre ma costante, perché uno stream audio/video con buchi nella consegna dei dati frequenti o molto lunghi diventa pressoché indecifrabile per l'utente che si trova all'altro capo della comunicazione. Se durante una telefonata con una persona si hanno dei periodi di silenzio dell'ordine di qualche secondo, anche nel caso in cui questi siano poco frequenti, comportano comunque un grave deterioramento nella qualitá del servizio, proprio perché ai secondi di silenzio seguirá un ulteriore periodo di inattivitá della comunicazione vera e propria, costituito da uno scambio di frasi tra i 2 interlocutori per riprendere la comunicazione, come "pronto; ci sei?; ripeti; non ho capito ...". Mostrato come puó essere deteriorante per una comunicazione VoIP un alto intervallo di indisponibilitá, un discorso analogo vale se la trasmissione é caratterizzata da un basso intervallo di continuitá. In questo caso infatti, durante la conversazione tra due persone, si avranno dei brevi ma frequenti periodi di silenzio che, in base alla durata, potrebbero rendere la comunicazione fastidiosa nel caso migliore (periodi di silenzio appena percettibili), oppure, nel caso pessimo, potrebbero rendere la comunicazione del tutto incomprensibile, proprio perché il nostro cervello non sarebbe in grado di ricostruire dai frammenti ascoltati le parole dette dall'altro interlocutore.

1.2.2 Problemi legati alla mobilitá

L'introduzione dell'elemento mobilitá nelle connessioni wireless ha portato al sorgere di nuove problematiche. Per poter accedere ad Internet ogni device deve ottenere un indirizzo IP che lo indentifichi nella rete. Di norma peró questo indirizzo ha validitá solo localmente, nella sottorete in cui si trova il device stesso. Il problema sorge quando la persona si sposta ed esce dal

raggio d'azione della sottorete di partenza. A questo punto risulta necessario collegarsi ad una sottorete diversa, che abbia copertura nel luogo in cui ci si trova. Con il passaggio peró da una sottorete ad un'altra, il device perderá il proprio indirizzo IP e dovrá ottenerne uno nuovo dalla nuova sottorete, evento che causerá l'interruzione di tutte le comunicazioni attive, a causa del fatto che tutte queste sessioni si riferiscono ad un indirizzo non piú utilizzato. Utilizzando tecnologie ad un raggio piú ampio, diminuiscono i casi in cui un device deve passare da una rete ad un'altra e quindi diminuisce la frequenza di interruzione delle comunicazioni. Anche per questo motivo sono nate le tecnologie ad ampio raggio come quelle di terza generazione. Purtroppo peró, tipicamente, man mano che si utilizza una tecnologia con un raggio d'azione maggiore, diminuiscono solitamente le prestazioni delle trasmissioni in termini di bandwidth (figura 1.7).

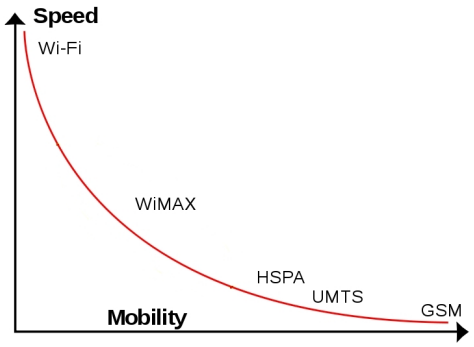

Figura 1.7: Mobilitá e bandwidth al variare della tecnologia

Handover

Preso atto del fatto che non esiste ad oggi una tecnologia perfetta che fornisca buon supporto alla mobilitá ed elevate prestazioni, negli ultimi anni si é cercato di risolvere questo problema facilitando il passaggio da una sottorete ad un'altra o da una tecnologia ad un'altra, passaggio detto handover. Esistono due tipologie di handover:

- *Handover orizzontale*: in questo caso il device passa da una sottorete ad un'altra utilizzando la stessa interfaccia di rete. Vi é un cambiamento di sottorete e quindi di indirizzo IP, utilizzando sempre la stessa tecnologia.

- *Handover verticale*: in questo caso vi é un contemporaneo cambiamento di sottorete e quindi di indirizzo, assieme ad un cambio di tecnologia utilizzata. Un device mobile puó possedere piú di un'interfaccia di rete: esistono in commercio per esempio smartphone che supportano connessioni 3G e WiFi, tablet con 3G e WiMax ... . Attualmente un utente, prima di connettersi ad Internet, sceglie quale tecnologia utilizzare e, nel caso in cui questa divenga successivamente indisponibile, si deve attivare per utilizzarne un'altra, perdendo peró le connessioni precedentemente instaurate. Nei sistemi invece che supportano l'handover verticale, sistemi in fase di progettazione e di sviluppo, ció non accade. Infatti, il passaggio da una tecnologia ad un'altra in questo caso non é piú traumatico, perché le connessioni instaurate rimangono attive di norma senza l'intervento dell'utente.

1.3 Apporto di HPforMSC

HPforMSC, il sistema da me progettatto ed oggetto di questa tesi ha come obiettivo il supporto all'handover orizzontale e verticale nei device mobili. Lo scopo di HPforMSC é quello di rendere trasparente all'utente la gestione delle varie interfacce di rete, sollevando l'utente dal carico della scelta dell'interfaccia di rete da utilizzare e da tutti i problemi che affliggono le comunicazioni in essere nel momento dell'handover, sia esso orizzontale oppure verticale. Attraverso HPforMSC l'utente puó sfruttare tutte le interfacce di rete in dotazione al device in uso. Di volta in volta, con granularitá al pacchetto, HPforMSC, in maniera del tutto trasparente, seleziona l'interfaccia di rete migliore, in base a criteri come i ritardi, le perdite dei messaggi, il costo per l'utente del traffico sull'interfaccia ...

Capitolo 2

Stato dell'arte

Gli approcci per la risoluzione del problema dell'handover orizzontale e verticale sono innumerevoli e in alcuni casi anche molto differenti tra di loro. Le varie soluzioni si differenziano per il livello nello stack di rete in cui agiscono, per la granularitá con cui é possibile decidere l'interfaccia di rete da usare, per le prestazioni ottenibili durante le fasi critiche di handover e per l'architettura necessaria per il supporto alle comunicazioni. L'obiettivo di tutte le soluzioni é quello di permettere ad un nodo mobile, da ora MN, la possibilitá di muoversi accedendo a reti differenti e ai relativi servizi, senza che l'utente possa accorgersi del cambio di rete e, nel caso, del cambio di tecnologia. Nelle comunicazioni basate su IP, l'indirizzo IP del nodo mobile svolge due ruoli differenti, identifica il nodo e fornisce informazioni sulla posizione del device nella rete. Quando il MN accede ad una rete, acquisisce un indirizzo IP valido localmente all'interno della rete stessa. Nel caso in cui il MN si muova e acceda ad una rete diversa, acquisirá un nuovo indirizzo IP, cambiando in questo modo la sua identitá, costituita in principio dall'indirizzo IP acquisito all'accesso alla prima rete. Per ogni comunicazione attiva, risulta quindi necessario comunicare all'altro end-point, detto CN (correspondent node), la nuova identitá, in modo da poter continuare la comunicazione. Ovviamente, anche il CN potrebbe, a sua volta, essere un nodo mobile ed andare in contro alle stesse problematiche.

2.1 Soluzione generale

La maggior parte delle architetture per la gestione della mobilitá ha una struttura comune, in quanto adotta meccanismi analoghi per la risoluzione di alcuni determinati problemi:

- Viene definito un identificativo univoco per i MN, indipendente dalla location del MN stesso, dalla rete a cui accede o dalla tecnologia che sta utilizzando.
- Viene realizzato un meccanismo di localizzazione dei MN, che mantenga l'associazione tra l'identificativo assegnato al MN e la sua posizione (tipicamente indicata tramite l'indirizzo IP attualmente in uso). Di norma l'architettura del gestore di mobilitá prevede la presenza di uno o piú nodi nella rete che svolgano questo compito, fornendo ai CN un servizio di localizzazione di MN sempre disponibile. Il principio generale di questo meccanismo prevede che inizialmente i MN si registrino su questi nodi, fornendo il proprio identificativo e l'indirizzo o gli indirizzi IP attualmente in uso; successivamente, ad ogni cambio di rete e quindi di IP, il MN ha il compito di notificare l'avvenuto al nodo su cui si era registrato, in modo che i CN possano, all'occorrenza, risalire alla posizione attuale del MN.

Ovviamente ció che distingue un'architettura da un'altra sono le modalitá con cui questi servizi vengono realizzati. Tipicamente si possono distinguere le varie soluzioni secondo differenti criteri:

- *Chi si occupa del servizio di localizzazione dei MN*:
  - *Soluzioni end-to-end pure*: il compito viene distribuito tra i due end-point della comunicazione.
  - *Soluzioni home-network based*: il compito viene svolto all'interno della rete a cui era connesso originalmente il MN.
  - *Soluzioni border gateway based*: il servizio viene mantenuto dai gateway di ogni rete attraversata dal MN.

  - *Soluzioni hybrid end-to-end*: il servizio di localizzazione risiede in uno o piú server, ma il compito di riconfigurazione dei nodi risiede nei due end-point della comunicazione.
  - *Soluzioni con external relay*: il compito di localizzazione e il servizio di supporto alla mobilitá viene svolto da un server separato, non facente parte né della rete di accesso dei MN, né delle reti attraversate dai MN. La presenza del server non va ad incidere sull'infrastruttura delle rete.

- *Granularitá*: La scelta dell'interfaccia di rete da utilizzare puó essere rivolta a tutte le comunicazioni attive sul MN, oppure puó essere fatta una scelta diversa per ogni flusso attivo nel MN; infine, alcuni approcci prevedono una granularitá ancora piú fine, quella al pacchetto, che per l'appunto prevede di poter effettuare scelte diverse sull'interfaccia da utilizzare per ogni datagram IP inviato.

- *Livello stack di rete*: La gestione della mobilitá puó coinvolgere diversi livelli dello stack di rete ISO-OSI, dal livello applicazione ai livelli transport, network, fisico. Seguendo questa classificazione, qui di seguito verranno descritte alcune soluzioni sviluppate ad oggi.

2.2 Soluzioni a livello Network

2.2.1 Sistemi basati su Mobile IPv6

- **Mobile IPv6**[3], **Fast Handover Mobile IPv6 (FMIP)**[4], **Hierarchical Mobile IPv6 (HMIP)**[5], **Proxy Mobile IPv6 (PMIP)**[6]: tutte queste soluzioni comportano l'utilizzo di un Home Agent (HA), un'entitá aggiuntiva inserita all'interno della rete locale a cui il MN appartiene originalmente. L'HA svolge la funzione di registro del MN e si occupa dell'inoltro dei pacchetti verso il MN quando quest'ultimo si trova in un'altra rete. Come indicato dal nome di queste soluzioni,

la rete deve supportare IPv6, in modo tale da sfruttare le estensioni dell'header dei datagram IPv6 per inserire gli identificativi del MN (l'indirizzo IP attuale) e del relativo HA. Un'altra limitazione legata a questo tipo di approcci é costituita dal fatto che, almeno attualmente, non é supportato l'utilizzo combinato di interfacce di rete multiple. Per ogni MN, viene registrato sull'HA un solo indirizzo IP, quindi relativo ad una sola interfaccia di rete. Studi ulteriori[7] hanno inoltre dimostrato che a causa dei numerosi messaggi necessari per la registrazione in caso di handover, la latenza in questa fase risulta essere abbastanza elevata. Ció ovviamente puó portare a disservizi in tutti quei servizi real-time, come per esempio il VoIP (Voice over IP). É possibile, tramite operazioni di return routability[1] che, previa lo scambio di messaggi con l'HA dopo un handover per conoscere i nuovi indirizzi IP, MN e CN dialoghino direttamente, in modo da abbassare il delay della comunicazione. Un problema nasce peró in caso di presenza di sistemi di NAT o di firewall, sistemi che rendono irraggiungibile il MN una volta che questo si sia spostato in un'altra rete e che non permettono la comunicazione diretta tra MN e CN, impedendo quindi il return routability.

- **Multiple Care of Address Registration (monami6)**[8]: questo sistema, basato su MIPv6, si caratterizza dal supportare il multihoming, permettendo cioé al MN la registrazione all'HA di molteplici indirizzi IP, associati ad una o piú interfacce di rete. Come per gli approcci precedenti, anche in questo caso si riscontrano gli stessi problemi nel caso di presenza di firewall o di sistemi di NAT.

- **Network Mobility Basic Support Protocol (NEMO BSP)**[2]: supporta la mobilitá di un'intera rete mobile, costituita da router mobili (MRs) e da "mobile network nodes" (MNNs). In questo caso non

[1] Processo attraverso il quale é possibile, grazie alla partecipazione di un nodo esterno, la comunicazione diretta tra MN e CN, anche dopo che MN e CN hanno perso le informazioni sull'indirizzo IP dell'altro end-point a causa di un handover.

é il singolo device a muoversi, ma sono tutti i nodi della rete che si muovono assieme. Questa caratteristica rende NEMO utilizzabile per esempio negli scenari di reti su mezzi di trasporto pubblico, come treni e autobus[9]. Il meccanismo di handover di MR é molto simile a quello del MN nei sistemi descritti in precedenza. MR, utilizzato come gateway, ottiene un indirizzo detto Home Address (HoA) dalla rete a cui appartiene inizialmente. Quando MR lascia la rete iniziale (home network) per entrare in un'altra rete (foreign network), ottiene un altro indirizzo, il Care of Address (CoA). Ad ogni nuovo CoA ottenuto, MR notifica il nuovo indirizzo al HA, in modo da mantenere la relazione tra HoA e CoA. In questo modo viene a crearsi un tunnel bidirezionale tra MR e HA. Tutti i pacchetti provenienti dal corresponding node avranno come destinazione HoA, saranno quindi ricevuti da HA, il quale li inoltrerá al CoA. Per quello che riguarda i MMN, questi ottengono un indirizzo permanente dal MR e tutti i messaggi originati o destinati ai MMN attraversano il tunnel descritto.

2.2.2 Location/ID Separation Protocol (LISP)

LISP[10],[11],[12] é una soluzione proposta da Cisco, del tipo border gateway. Prevede l'aggiunta di un livello LISP allo stack di rete dei router che mettono in comunicazione la sottorete con l'esterno. Questi router svolgono il ruolo di Ingress Tunnel Router (ITR) e di Egress Tunnel Router (ETR). Gli ITR intercettano i datagram IP provenienti dal MN, creano un'associazione tra l'indirizzo del nodo mobile ed il suo id, aggiungono un header LISP al pacchetto e lo inoltrano alla destinazione attraverso i router LISP. Quando un ETR appartenente alla stessa rete a cui appartiene il nodo destinazione riceve un pacchetto LISP, estrae il datagram IP dal pacchetto, memorizza l'associazione tra l'id destinazione e il suo indirizzo IP ed inoltra il datagram a destinazione. Un tale approccio non comporta modifiche degli end-point della comunicazione, ma necessita che le due reti abbiano sui loro confini dei router LISP che agiscano come ITR e ETR.

2.3 Soluzioni tra livello Network e Transport

Queste soluzione sono caratterizzate dall'inserimento di un livello intermedio tra i livelli network e transport dello stack di rete di ogni nodo presente ai due capi della comunicazione (end-nodes). Soluzioni che seguono questo approccio sono per esempio Host Identity Protocol (HIP)[13], Location Independent Addressing for IPv6 (LIN6)[14] e Level 3 Multihoming Shim Protocol for IPv6 (Shim6)[15]. Il nodo che svolge il compito di location register é molto simile ad un server DNS, in quanto opera come servizio esterno alla rete di accesso che mantiene l'associazione tra l'identificativo di un MN e la sua locazione. Una limitazione presente in questi approcci é la necessitá di modificare lo stack di rete di tutti gli end-nodes, sia nel caso in cui essi siano nodi mobili, sia nel caso in cui uno di questi (il CN) sia un nodo fisso.

2.4 Soluzioni a livello Transport

Agiscono a questo livello il Datagram Congestion Control Protocol (DCCP)[16] e il Mobile Stream Control Transport Protocol (m-SCTP)[17]. Mentre per quello che riguarda la granularitá delle due soluzioni abbiamo delle differenze, in quanto la prima é orientata al datagram mentre la seconda é orientata allo stream, entrambe mantengono lo stesso approccio: gli end-system svolgono il ruolo di location register, informando in modo proattivo[2] il CN ogni qualvolta la configurazione IP del MN cambia (il MN si é associato con una nuova rete ed ha quindi cambiato indirizzo IP). Quando peró entrambi gli end-nodes sono dei nodi mobili, é possibile che le configurazioni di entrambi i nodi cambino simultaneamente[3]; in questo caso ogni nodo possiede informazioni vecchie e non piú valide sull'altro end-node, rendendo quindi

[2]Lo scambio delle informazioni avviene periodicamente oppure quando si ha un cambiamento nello stato. Non é necessaria la richiesta di informazioni da parte dell'altro end-node

[3]Per cambiamento simultaneo si intende che entrambi i nodi cambiano la propria configurazione prima di ricevere la nuova configurazione dall'altro end-node.

mutualmente irraggiungibili i due nodi. Anche in questo caso, le soluzioni a livello transport hanno il limite di richiedere la modifica dello stack di rete a tutti gli end-nodes delle comunicazioni, siano essi mobili oppure fissi.

2.5 Soluzioni a livello Session

A questo livello la maggioranza delle soluzioni al problema della mobilitá si basano sul protocollo SIP (Session Initiation Protocol), un protocollo di sessione utilizzato principalmente per il controllo delle comunicazioni multimediali tra due end-nodes[18],[19].

- TMSP (Terminal Mobility Support Protocol)[20] utilizza un server SIP esterno alla rete di accesso come location register, il quale mantiene l'associazione tra la locazione del nodo mobile (l'indirizzo IP) e il suo identificativo utente (per esempio grassig@cs.unibo.it). TMSP sfrutta messaggi di REGISTER e INVITE del protocollo SIP per la gestione della mobilitá; in SIP il messaggio INVITE viene utilizzato per iniziare una comunicazione tra due nodi, mentre il messaggio REGISTER ha il compito di comunicare agli altri nodi la disponibilitá alla comunicazione. In ambito mobile, TMSP prevede che il MN invii un messaggio REGISTER al server SIP per aggiornare la propria posizione, mentre il messaggio INVITE viene utilizzato per stabilire la comunicazione diretta con un altro nodo.

- NetCAPE (Enabling Seamless IMS Service Delivery across Heterogeneous Mobile Networks)[21]: prevede la gestione dell'handover verticale, mediante l'utilizzo di un livello sessione simile a SIP e detto IP Multimedia Subsystem, il quale ha il compito di gestire la sessione di comunicazione e i processi di autenticazione di tutte le reti IP a cui é connesso il nodo.

2.6 Soluzioni ai problemi di NAT e Firewall

La maggior parte delle soluzioni descritte in precedenza non tengono conto della possibile presenza di sistemi di firewall o NAT nei due end-nodes, che possono impedire la comunicazione tra MN e CN quando il MN cambia indirizzo IP. Tipicamente, quando uno dei due nodi si trova all'interno di un sistema NAT o dotato di firewall, viene utilizzato un server esterno STUN[22] o TURN[23], in modo da ovviare a questo problema. Quando peró a trovarsi in sistemi con NAT o firewall sono entrambi gli end-nodes di una comunicazione, questo approccio risulta inutile e le soluzioni descritte in precedenza diventano vane. In questi casi é spesso necessario l'utilizzo di un server esterno ai firewall che faccia da relay a livello applicazione, ricevendo ed inoltrando tutti i pacchetti che i due nodi si scambiano durante tutta la comunicazione.
MMUSE[24] é un sistema che necessita di un server SIP ausiliare posizionato al confine del sistema nel quale il MN si muove. Questo sistema puó essere composto da numerose subnets, non necessariamente associate ad una tecnologia comune. Il MN puó muoversi liberamente all'interno del sistema, da una sottorete ad un'altra e quindi cambiando indirizzo IP; della sua posizione viene tenuta traccia dal server SIP, detto SBC (Session Border Controller). SBC svolge contemporaneamente il ruolo di server SIP, proxy RTP, firewall e NAT, intercettando tutte le comunicazioni che entrano o escono dal sistema. Mediante protocollo SIP MN e CN possono richiedere a SBC di permettere la comunicazione tra i due nodi. Nel caso in cui il MN si sposti in una subnet diversa, cambiando quindi indirizzo IP, SBC si occuperá di mascherare questo cambiamento di configurazione al CN.
La limitazione di MMUSE é costituita principalmente dal fatto che tutto il traffico deve passare dal SBC posto al confine del sistema in cui si trova il MN e che lo stesso MN si puó muovere solamente in quelle subnets interne al sistema controllato dal SBC.

2.7 Soluzioni che non richiedono la modifica della rete

Alcune delle soluzioni descritte in precedenza richiedono la modifica di hardware e/o software di nodi interni alla rete, limite che di fatto le ha rese fino ad oggi pressoché inutilizzate. Al contrario, esiste un'intera classe di soluzioni che, proprio per evitare questo problema, utilizza un approccio diverso, impiegando un relay esterno alla rete per supportare la mobilitá. Questo approccio prevede la divisione della comunicazione tra MN e CN in due parti separate: da un lato vi é il MN e il relay, dall'altro il relay che comunica con il CN. Il ruolo del relay é quindi quello di fare da tramite tra MN e CN in tutte le loro comunicazioni. Ció comporta allo stesso tempo sia la mancanza della necessitá di dover cambiare i nodi interni della rete, sia la risoluzione del problema dei sistemi NAT e Firewall, visti in precedenza.

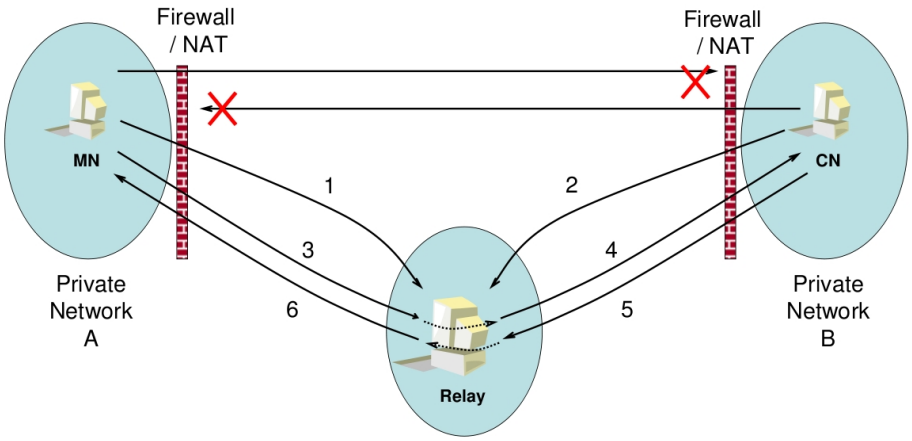

Figura 2.1: Il data Relay permette la comunicazione indiretta tra i due nodi che si trovano in sistemi con firewall - NAT

La classe delle soluzioni che prevedono l'utilizzo di un relay esterno puó essere a sua volta suddivisa in due classi distinte, gli approcci visibili e quelli invisibili:

- *Relay visibili*: Questa classe supporta solo quelle applicazioni e quei protocolli che prevedono l'utilizzo esplicito di una coppia di proxy, il client sul MN e il server sul relay. La scelta su quale proxy server utilizzare viene fatta direttamente dalle applicazioni sul MN. Un approccio di questo tipo é presente in Always Best Packet Switching (ABPS-SIP/RTP)[25], un'architettura che supporta la mobilitá per applicazioni basate su SIP/RTP (VoIP, VOD ...). ABPS-SIP/RTP opera a livello sessione e prevede la presenza di un proxy client sul MN e di un proxy server in un server esterno alla rete, un relay. Il MN puó spostarsi su reti differenti, mentre il proxy client utilizza una tecnica cross-layer per monitorare lo stato e le performance di tutte le interfacce di rete attive per determinare, per ogni datagram IP inviato, quale sia l'interfaccia migliore da utilizzare. Ogni qualvolta un'interfaccia di rete diventi attiva oppure inattiva, é lo stesso proxy client ad occuparsi della riconfigurazione del sistema. Un importante caratteristica che differenzia ABPS-SIP/RTP da tutti i sistemi descritti in precedenza consiste nella scelta dell'interfaccia di rete da utilizzare: mentre le soluzioni precedenti che supportano piú interfacce di rete passano da un'interfaccia all'altra solo quando la prima diventa inattiva, ABPS-SIP/RTP non attende l'inattivitá dell'interfaccia, ma sceglie sempre quella con le migliori prestazioni. Un limite di ABPS-SIP/RTP sta nel fatto che non é una soluzione general purpose, ma supporta solamente applicazioni basate su SIP/RTP.

- *Relay invisibili*: a differenza della precedente, questa classe supporta anche quelle applicazioni e quei protocolli che non prevedono l'utilizzo esplicito di un proxy, creando una sorta di tunnel tra MN e il proxy server nascosto alle applicazioni sul MN. Dall'altro lato, il CN considera il proxy server come se fosse lui stesso il MN. Il proxy client che risiede nel MN cattura il traffico in uscita e lo redirige verso il proxy server, senza che le applicazioni sul MN si accorgano di nulla. Tipicamente viene creata sul MN un'interfaccia di rete virtuale, ossia non connessa

direttamente ad un'interfaccia fisica, e viene modificata la tabella di routing in modo tale da farla utilizzare da tutte le applicazioni presenti nel MN. In questo modo il proxy client puó catturare tutto il traffico passante per questa interfaccia virtuale ed inviarlo verso il proxy server utilizzando l'interfaccia fisica che preferisce. Una soluzione appartentente a questa classe é [26], un'architettura ad hidden proxy, la quale si basa su meccanismi quali tun/tap e iptable/netfilter per catturare il traffico sul MN e redirigerlo verso il proxy server. Limite a [26] é il fatto che supporta solamente connessioni TCP.

2.8 Architetture per il supporto alla mobilitá a confronto

Vengono qui di seguito riportate delle tabelle riassuntive delle caratteristiche principali di varie architetture che supportano la mobilitá. Piú in specifico, le varie caratteristiche vengono suddivise nelle seguenti classi:

- *Requisiti*: specifica quali protocolli siano necessari;

	Classi	end-to-end			end-to-end ibridi					home network				Access network		relay invisibili	relay visibili
		TCP-migrate	DCCP	m-SCTP	HIP	LIN6	Shim6	TMSP	HIMAS	MIPV6...	Monami6	Nemo	FlowMob	LISP	MMUSE	HiddenProxy	ABPS
Requisiti	IPv6					Y	Y			Y	Y	Y	Y	Y			
	MIPv6									Y	Y	Y	Y				
	IPSec									Y	Y	Y	Y				

Tabella 2.1: Architetture per il supporto alla mobilitá a confronto, requisiti

- *Deployment*: indica quali modifiche é necessario applicare ai nodi della rete, da MN e CN a router, gateway ... Mentre modifiche allo stack di rete del MN non comportano grandi inconvenienti perché questi cambiamenti portano un diretto vantaggio al MN stesso, che é quindi moti-

vato a farli, lo stesso discorso non é piú valido quando le modifiche interessano lo stack protocollare di CN. Infatti questo potrebbe essere un nodo fisso, quindi tutti i cambiamenti comporterebbero un costo, non sempre trascurabile, senza peró il necessario tornaconto, come nel caso dei MN, proprio perché il poter supportare la mobilitá non porta alcun miglioramento nelle performance dei nodi fissi, ma solo costi aggiuntivi. Il discorso rimane simile quando i requisiti coinvolgono la modifica delle applicazioni. In questo caso peró anche nel MN ci possono essere dei problemi, legati soprattutto alle applicazioni legacy, difficilmente modificabili e quindi inutilizzabili se non rispettano i nuovi requisiti posti dal sistema per la gestione della mobilitá. Altri requisiti tipici di questi sistemi coinvolgono i nodi interni della rete, quali router, access point, gateway... Solitamente le architetture che richiedono queste modifiche, soprattutto quando coinvolgono molti nodi di questo tipo, si vanno a scontrare con una certa inerzia nel deployment della soluzione stessa, proprio perché queste modifiche possono richiedere investimenti considerevoli per le compagnie che vogliono offrire questo tipo di servizio per la mobilitá. Al contrario, come visto in precedenza nella sezione dei relay esterni, quando l'architettura prevede la presenza di un proxy server esterno alla rete, senza richiedere la modifica dei nodi interni, il suo sviluppo diventa piú facilmente attuabile in termini economici.

- *Performance*: descrive la qualitá dei servizi offerti dall'architettura, sia in termini di applicazioni e protocolli supportati (TCP, UDP, SIP ...), sia in termini di continuitá del servizio, latenza nella comunicazione, granularitá nella scelta dell'interfaccia di rete da usare, possibilitá di utilizzare una o piú interfacce contemporaneamente. Vengono inoltre valutati:

  - *Intervallo di indisponibilitá*: misura l'intervallo di tempo durante il quale il MN non puó comunicare con il CN a causa di un handover. Questo elemento risulta essere fondamentale per valutare la

bontá del sistema nell'essere utilizzato da applicazioni interattive, come per esempio il VoIP, dove un lungo intervallo di indisponibilitá porterebbe a gravi disservizi. Questo intervallo racchiude il tempo necessario per riconfigurare tutti i nodi della rete coinvolti nella comunicazione tra MN e CN, piú l'intervallo di tempo che precede un handover, caratterizzato tipicamente da un'elevata percentuale di perdita di pacchetti.

– *Intervallo di continuitá*: misura l'intervallo di tempo durante il quale MN e CN possono comunicare normalmente; in altre parole indica la distanza tra un intervallo di indisponibilitá e l'altro.

– *Latenza end-to-end*: misura il tempo necessario per consegnare un messaggio dal MN al CN.

	Classi	end-to-end			end-to-end ibridi					home network				Access network		relay invisibili	relay visibili
		TCP-migrate	DCCP	m-SCTP	HIP	LIN6	Shim6	TMSP	HIMAS	MIPV6...	Monami6	Nemo	FlowMob	LISP	MMUSE	HiddenProxy	ABPS
Modifiche necessarie	Protocol stack in MN	Y	Y	Y	Y	Y	Y			Y	Y	Y	Y	Y	Y	Y	
	Protocol stack in CN	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y				
	Access network									Y	Y	Y					
	Border Gateway													Y	Y		
	External Relay															Y	Y
	Applicazione in MN		Y	Y				Y	Y								
	Applicazione in CN		Y	Y				Y	Y								

Tabella 2.2: Architetture per il supporto alla mobilitá a confronto, modifiche necessarie

	Classi	end-to-end			end-to-end ibridi					home network				Access network		relay invisibili	relay visibili
		TCP-migrate	DCCP	m-SCTP	HIP	LIN6	Shim6	TMSP	HIMAS	MIPV6...	Monami6	Nemo	FlowMob	LISP	MMUSE	HiddenProxy	ABPS
Performance	Supporto a SIP/RTP				Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y		Y
	Supporto a TCP	Y			Y	Y	Y		Y	Y	Y	Y	Y	Y		Y	Y
	Supporto a UDP				Y	Y	Y		Y	Y	Y	Y	Y	Y			
	Supporto a app. legacy								Y	Y	Y	Y		Y		Y	
	Identificazione sender									Y	Y	Y		Y			Y
	NIC multipli simultanei										Y	Y	Y	Y		Y	Y
	Granularitá	C	C	C	N	N	N	N	N	N	N	N	C	C	C	P	P
	No limitazione FW e NAT														Y	Y	Y
	Intervallo di indisponibilitá durante handover	M ∞	M ∞	M ∞	M	M	M	A	A	A	A	A	A	M ∞	A ∞	M	B
	Intervallo di continuitá	B	B	B	B	B	B	B	B	B	A	A	A	M	M	A	A
	Latenza end-to-end (caso ottimo)[2]	B **	B **	B **	B **	B **	B **	B **	B **	B-A ***	B-A ** *	B-A ** *	B **	M **	M-A ***	M-A ***	M-A ***
	Latenza end-to-end (caso medio)[1]	A	A	A	A	A	A	A	A	VA	VA	VA	VA	VA	M-A	M-A	M-A ***

[1] Scenario con l'introduzione di un relay a causa di FW e NAT.

[2] Scenario senza l'introduzione di un relay.

[*] La latenza diventa alta se il processo di return routability fallisce a causa della presenza di firewall.

[**] La latenza diventa alta se, a causa di un firewall, occorre introdurre un relay esterno.

[***] La latenza diventa alta se il relay (server) é lontano dal MN.

Tabella 2.3: Architetture per il supporto alla mobilitá a confronto, performance

Come leggere la tabella: Per quello che riguarda l'intervallo di indisponibilitá, si considera:

- *A: alto* se oltre a riconfigurare l'interfaccia occorre autenticarsi nuovamente su un server (o home agent) e quindi attendere lo scambio di messaggi tra MN e HA e tra HA e CN.

- *M: medio* se oltre a riconfigurare l'interfaccia occorre solamente inviare un messaggio verso un nodo specifico (border gateway, relay o server ed eventualmente attendere la risposta.

- *B: basso* se oltre a riconfigurare l'interfaccia occorre solamente inviare un messaggio ad un nodo specifico (border gateway, relay o server) senza dover attendere una risposta o se l'architettura é in grado di individuare il momento in cui si sta iniziando a perdere pacchetti e quindi di inviarli nuovamente su un'altra interfaccia.

- ∞ se é possibile che il sistema non riesca a ripristinare la connettivitá.

Per quello che riguarda la latenza end-to-end invece si considera:

- *B: basso* se la comunicazione é diretta (vi é compreso il return routability se ha successo) o se il nodo intermedio é il border gateway, che é quindi giá presente nella route.

- *M: medio* se la comunicazione deve passare per un home agent.

- *A: alto* se la comunicazione deve passare per un nodo esterno lontano dalla route.

2.8.1 Analisi tabella

Dalla tabella 2.3 é possibile ricavare alcune importanti informazioni sulle prestazioni delle varie tipologie di architetture:

- *Intervallo di indisponibilitá*: Tutte le architetture basate su MIPv6 presentano un elevato intervallo di indisponibilitá[27],[7], anche dell'ordine di 1-3 secondi. Questo perché quando avviene un handover i protocolli richiedono un intenso scambio di messaggi tra MN, la sua home network e il CN, in modo da registrare il nuovo indirizzo all'home agent e permettere il ritorno di routability. Tuttavia, se durante la fase di handover il device puó utilizzare un'altra interfaccia di rete, il tempo di indisponibilitá della comunicazione puó diminuire. Questo spiega perché le architetture che prevedono multihoming a livello di pacchetto sono caratterizzate da un intervallo di indisponibilitá piú basso. Inoltre, se, come nel caso di ABPS, l'architettura prevede il monitoraggio dello stato del canale per l'identificazione dei pacchetti persi e la possibilitá di cambiare l'interfaccia in uso prima che questa diventi inutilizzabile, é possibile diminuire la perdita di pacchetti che solitamente si ha nella fase che precede l'handover, diminuendo in questo modo l'intervallo di indisponibilitá.

- *Intervallo di continuitá*: le architetture che non prevedono la possibilitá di utilizzare simultaneamente diverse interfacce di rete sono caratterizzate da un basso intervallo di continuitá. Questo perché l'unico metodo per non avere un'interruzione nella comunicazione durante un handover é il poter utilizzare un'altra interfaccia di rete giá attiva e configurata. Al contrario, le soluzioni che prevedono il monitoraggio dello stato del canale assieme al multihoming presentano un elevato intervallo di continuitá grazie al fatto che rendono possibile l'individuazione tempestiva di un pacchetto perso e la sua ritrasmissione attraverso un'altra interfaccia.

- *Latenza end-to-end*: Le soluzioni che prevedono una comunicazione diretta tra MN e CN (end-to-end pure o ibride) sono caratterizzate da una bassa latenza, perché non vi sono intermediari nella comunicazione che deviano il percorso dei pacchetti. Anche le architetture di tipo ac-

cess network presentano bassi delay, nonostante il fatto che introducano un intermediario nella comunicazione tra MN e CN. Il motivo risiede nel fatto che questo punto intermedio é il border gateway, giá presente comunque nel path tra MN e CN. Per quello che riguarda le architetture di tipo home network, nel caso in cui la fase di return routability abbia successo, il delay é basso perché si ha una comunicazione diretta tra MN e CN. Nel caso in cui invece la return routability fallisce, tutti i pacchetti devono passare per l'home network, comportando un aumento del delay, aumento che dipenderá dalla distanza tra l'home network e i due end-node. Infine, le architetture che prevedono l'utilizzo di un relay esterno sono caratterizzate da delay maggiori, proprio perché le comunicazioni tra MN e CN subiscono una deviazione e sono obbligate a passare per il relay. Fatta questa premessa, occorre peró porre l'attenzione sul fatto che in molti casi si ha la presenza di un firewall o di un sistema di NAT, i quali comunque, indipendentemente dall'architettura per la gestione della mobilitá utilizzata, non permettono la comunicazione diretta tra MN e CN, ma obbligano tutto il traffico a passare per un relay. In questo caso anche i valori di delay delle architetture che non prevedono l'utilizzo di un relay per la mobilitá crescono notevolmente.

Capitolo 3

Obiettivi

L'obiettivo di HPforMSC é quello di fornire un supporto alla mobilitá di dispositivi wireless riducendo al minimo i periodi di discontinuitá del servizio di connettivitá. L'architettura realizzata si pone lo scopo di sfruttare appieno tutte le interfacce di rete disponibili, escludendo l'utente dagli oneri della gestione contemporanea delle molteplici interfacce wireless e dei relativi handover. L'utente non deve rendersi conto del passaggio da un'interfaccia di rete all'altra, cambiamento che deve quindi essere il piú possibile impercettibile: tutti i servizi attivi sul device e che sfruttano una connessione ad Internet non devono venir influenzati dal funzionamento di HPforMSC e dalle sue scelte in termini di interfaccia da utilizzare. Queste scelte rispettano molteplici criteri, dalla qualitá della connessione al costo economico che l'utente deve sostenere per l'utilizzo.

3.1 Problemi connessi con IP

Come descritto in precedenza, l'introduzione del fattore mobilitá ha introdotto nuove problematiche nel mondo delle comunicazioni wireless. Mentre la tecnologia permette all'utente di muoversi liberamente utilizzando i propri dispositivi mobili, i protocolli ad oggi in utilizzo per la comunicazione su Internet comportano ancora dei freni alla libertá di movimento. Le co-

municazioni sulla rete Internet sono regolamentate dal protocollo IP, il cui funzionamento per l'instradamento dei pacchetti si basa sull'utilizzo dell'indirizzo IP sia come identificativo del device che come informazione per la localizzazione del device stesso. Il problema legato alla mobilitá consiste nel fatto che i device cambiano posizione[1], quindi si rende necessario cambiare l'informazione sulla localizzazione del device, cosa che peró comporta se si utilizza IP anche il cambiamento di identificativo. Ovviamente, se un identificativo puó variare nel tempo, diventa del tutto inutile ed é per questo motivo che occorre individuare un sistema alternativo da affiancare ad IP per gestire in maniera separata l'informazione sulla posizione di un device e il suo identificativo, in modo che quest'ultimo rimanga costante per tutto il tempo in cui il device é connesso ad Internet.

3.2 Mancanze soluzioni attuali

Come visto nel capitolo Stato dell'arte, non esiste una soluzione ottimale. Ad oggi, ogni architettura realizzata é caratterizzata dall'avere almeno un punto debole. Tutte le soluzioni che richiedono modifiche sul CN, sia a livello applicazione che all'interno dello stack di rete, si scontrano con la volontá del CN di non fare modifiche perché queste comportano un aumento dei costi e sono superflue se il CN é un nodo fisso. Discorso simile per quello che riguarda le architetture che richiedono modifiche ai nodi della home network del MN. Infatti, anche in questo caso, il problema é economico. Approcci di questo genere infatti richiederebbero la modifica di ogni sottorete esistente ad oggi che voglia fornire supporto alla mobilitá. In tutti questi casi il problema piú grande é la scarsa fattibilitá del deployment di queste soluzioni.

Oltre ai limiti di fattibilitá di alcuni tipi di approcci, le soluzioni attuali hanno, come osservabile anche dalla tabella 2.3, alcuni limiti legati alle per-

[1]Con cambiamento della posizione si intende lo spostamento fisico del device associato al cambiamento di rete e quindi di indirizzo IP

formance ottenibili dal sistema. Alcune di queste soluzioni mancano di generalitá, essendo utilizzabili solo con alcuni protocolli (SIP, TCP o UDP). Ovviamente questo é un forte limite se si vuol realizzare un'architettura completa per la gestione della mobilitá per ogni tipo di connessione, ma possono comunque essere utili per lo sviluppo di soluzioni ad-hoc per il supporto alla mobilitá di una predefinita cerchia di servizi. Un altro limite non trascurabile é senz'altro l'interazione che esiste tra l'architettura per la gestione della mobilitá e le applicazioni che ne trarranno profitto sul MN. Mentre alcune soluzioni sono del tutto trasparenti alle applicazioni, altre invece richiedono caratteristiche ben precise alle applicazioni sul MN[2], come il supporto all'utilizzo di proxy nella comunicazione. Nonostante sia possibile sostenere che modifiche sul MN siano maggiormente fattibili, essendo il MN il diretto interessato e beneficiario del supporto alla mobilitá, porre delle restrizioni sulle applicazioni comporta comunque un notevole sforzo per poter adattare tutte le applicazioni, anche non molto recenti, ai nuovi requisiti. É vero comunque che, come nel caso delle limitazioni riguardanti i protocolli supportati, é possibile vedere questi tipi di approcci come soluzioni orientate al supporto della mobilitá al singolo servizio o applicazione e non a soluzioni general purpose.

Per quello che riguarda il supporto all'utilizzo simultaneo di piú interfacce di rete, alcune soluzioni non prevedono questa possibilitá. Questo é un limite alla bontá dell'architettura principalmente per due motivi:

- Non poter utilizzare per la stessa comunicazione due interfacce diverse, benché il device in uso ne sia dotato, significa non sfruttare al massimo le potenzialitá dell'hardware in dotazione.

- Non é possibile mitigare i limiti di una tecnologia wireless con i pregi di una tecnologia diversa utilizzando contemporaneamente piú interfacce di rete, quindi non é possibile sfruttare l'elevata banda del WiFi assieme per esempio a tecnologie 3G caratterizzate invece da un raggio

[2]Vedi supporto alle applicazioni legacy della tabella 2.3

di copertura maggiore e quindi un intrinseco supporto alla mobilitá[3]. Perció in architetture di questo tipo l'utente deve fare una scelta a priori sull'interfaccia di rete da utilizzare, mediando per esempio tra il costo in denaro di ogni interfaccia, la banda ottenibile e la copertura della tecnologia: per esempio in device dotati di 3G e WiFi, l'utente dovrá scegliere se utilizzare una tecnologia ad ampio raggio ma con minor bandwidth e tipicamente con un maggior costo come 3G, oppure preferire il WiFi, tecnologia piú a buon mercato e con prestazioni migliori, ma con una copertura tipicamente inferiore al 3G e quindi con maggiori problemi in caso di mobilitá. Ció che invece risulta possibile quando l'architettura supporta il multihoming é il poter scegliere dinamicamente l'interfaccia da usare, per esempio favorendo l'utilizzo di tecnologie come il WiFi quando vi é copertura, per poi passare a 3G o WiMax, tecnologie a raggio maggiore, quando la copertura WiFi é assente.

3.2.1 Limiti prestazionali

Molte delle architetture descritte in precedenza non supportano nella versione originale la presenza di sistemi Firewall o di NAT. In questi casi, le architetture devono essere adattate inserendo un nodo esterno, un relay, fattore che comporta un peggioramento delle prestazioni per quanto riguarda il delay end-to-end della comunicazione e un aumento dei costi. Un altro elemento molto importante nella valutazione di un'architettura per il supporto alla mobilitá é la granularitá della scelta dell'interfaccia da utilizzare. Come visto nella tabella 2.3, esistono tre livelli di granularitá, quella per nodo, per canale e per pacchetto. Il livello migliore, non supportato da tutte le soluzioni, é senza dubbio l'ultimo, la granularitá per pacchetto. Questo perché piú é piccola la granularitá, piú efficacemente é possibile rispondere ai cambiamenti dello stato delle comunicazioni variando l'interfaccia da usare

[3]Vedi figura 1.7: rapporto tra mobilitá e banda delle varie tecnologie.

o la rete a cui si é associati, diminuendo quindi il calo di prestazioni durante l'handover.

3.3 Apporto di HPforMSC

Ció che HPforMSC introduce nello scenario dei gestori di mobilitá é un'architettura basata sull'utilizzo di un relay esterno al supporto della comunicazione tra MN e CN. HPforMSC supporta l'utilizzo contemporaneo di interfacce di rete molteplici e di differente tecnologia (WiFi, 3G, WiMax), con una granularitá nella scelta dell'interfaccia da usare a livello di datagram IP. Utilizzando HPforMSC non risulta necessario modificare alcun nodo della rete, come nei casi delle soluzioni basate su MobileIP. Il relay, costituito da un hidden proxy, é esterno alla rete e, purché all'interno di Internet, puó essere situato in qualsiasi luogo. Oltre a non comportare modifiche ai nodi interni delle reti attraversate dal MN, HPforMSC lascia inalterato anche il CN, il quale rimane del tutto estraneo alla gestione della mobilitá del MN, unico nodo, assieme al relay, a dover subire cambiamenti. Questi cambiamenti peró, a differenza di altre architetture, non intaccano il sistema operativo, ma rimangono tutte a livello applicazione, in modo da rendere HPforMSC separato dall'evoluzione del kernel in uso dal MN[4]. Tutti questi fattori rendono HPforMSC facilmente sviluppabile nel mondo reale, in quanto non comporta alcuna modifica né ai nodi della rete né al CN, ma necessita solo l'introduzione di un proxy server accessibile su Internet.
A differenza di altre architetture, HPforMSC supporta molteplici protocolli di trasporto, da UDP a TCP a ICMP; inoltre, utilizzando un hidden proxy, non comporta alcuna restrizione alle applicazioni in esecuzione sul MN, perché per l'appunto il proxy server é nascosto alle applicazioni stesse, le quali quindi rimangono del tutto estranee alla gestione della mobilitá, rendendo quindi compatibile HPforMSC con tutte le applicazioni, anche quelle legacy e che non supportano l'utilizzo di proxy espliciti.

[4]HPforMSC é stato attualmente realizzato solo per sistemi Linux

Inoltre, utilizzando un relay esterno, HPforMSC non é soggetto ai problemi legati alla presenza di Firewall o sistemi NAT che nascondono il MN al resto di Internet.
Per quello che riguarda le prestazioni ottenibili utilizzando HPforMSC, sfruttando simultaneamente tutte le interfacce di rete disponibili, é possibile ottenere intervalli di indisponibilitá della connettivitá molto bassi, proprio perché quando un'interfaccia inizia ad avere prestazioni non soddisfacenti, HPforMSC permette al MN di utilizzare le altre interfacce di rete, in modo da superare i disservizi della prima interfaccia. Per lo stesso motivo, l'intervallo di continuitá della comunicazione risulta essere abbastanza elevato, grazie al fatto che quando sta avvenendo un handover in un'interfaccia di rete, é possibile utilizzarne un'altra senza ripercussioni sulla qualitá della comunicazione stessa.
La politica di scelta dell'interfaccia seguita da HPforMSC prevede, per ogni datagram IP da inviare, di utilizzare sempre l'interfaccia di rete che garantisce la miglior qualitá del canale, intesa sia come delay della trasmissione che come numero di pacchetti persi. Questa politica puó essere integrata anche con il costo dell'utilizzo dell'interfaccia da parte dell'utente e il tipo di applicazione in uso, modificando eventualmente anche di volta in volta i pesi di questi elementi nel computo finale della qualitá di ogni interfaccia.

Capitolo 4

Strumenti

4.1 Routing nei sistemi Linux [1]

Protocolli di livello 3 dello stack di rete, come IP, devono occuparsi di individuare la via piú effficiente per permettere ad un messaggio di raggiungere la destinazione, ovunque essa sia (nella stessa sottorete o dalla parte opposta del mondo). Il processo che si occupa di definire questa strada e di individuare l'hop successivo a cui inoltrare un pacchetto viene detto *routing*. Il cuore del processo di routing si basa su 3 elementi, gli *indirizzi*, le *routes* e le *regole*.

- Indirizzo: definisce la posizione di un "servizio" nella rete.
- Route: definisce il percorso (solitamente solo il next hop) per raggiungere un determinato indirizzo.
- Regole: specificano i vari casi d'uso delle route e degli eventuali filtri da applicare al traffico in uscita ed in entrata al nodo.

Le informazioni necessarie per decidere se un pacchetto in entrata sia indirizzato al local host oppure debba essere inoltrato ad un altro destinatario, assieme alle informazioni necessarie per inoltrare il messaggio, sono contenute in un database detto Forwarding Information Base (FIB), piú comunemente

conosciuto come tabella di routing. Ogni nodo della rete, indipendentemente dal suo ruolo, possiede una tabella di routing che consulta ogni qualvolta sia necessaria la gestione di un messaggio in entrata o in uscita.

4.1.1 Regole di routing

La tabella di routing é una collezione di route. A sua volta, una route é una collezione di parametri che codificano le informazioni necessarie per poter inoltrare correttamente un messaggio a destinazione. Tra questi parametri, i principali sono:

- *Destination network*: indica l'indirizzo della rete in cui si trova la destinazione.
- *Egress device*: indica l'interfaccia che dovrá essere utilizzata per inviare il pacchetto in uscita.
- *Next hop gateway*: é l'indirizzo del router al quale dovrá essere inviato il messaggio nel caso in cui la destination network non sia direttamente connessa con l'host.

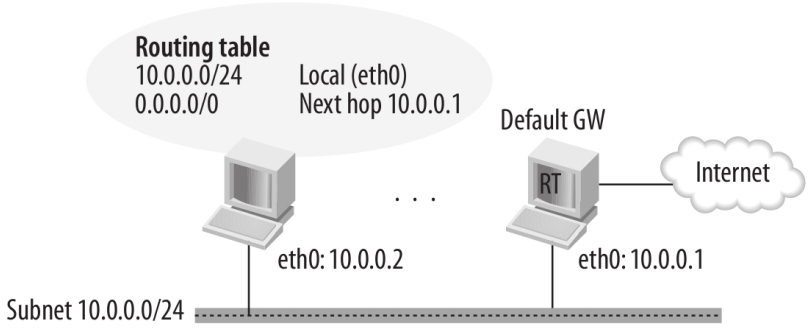

Figura 4.1: Esempio tabelle di routing e router[1]

Scope delle regole e degli indirizzi Lo scope di una regola é un indicatore della distanza dalla destination network, mentre lo scope di un indirizzo

IP indica quanto distante sia l'indirizzo dal local host. I valori di scope possibili sono i seguenti:

- Host: quando si tratta di indirizzi, indica che si tratta di un indirizzo locale dell'host stesso (per esempio 127.0.0.1); se si tratta di route, indica che la route porta ad un indirizzo del local host.
- Link: un indirizzo IP ha scope link se si riferisce ad una stessa LAN. Nel caso delle route, significa che la route indica il cammino per raggiungere un indirizzo della rete locale in cui si trova lo stesso host.
- Universe: é lo scope di tutti quegli indirizzi IP che non ricadono nei casi precedenti. Una route con scope universe porta ad un indirizzo che si trova ad una distanza maggiore di un singolo hop.

Quando un host deve inviare un pacchetto, il kernel deve individuare l'indirizzo IP da utilizzare come indirizzo sorgente. Se l'host possiede piú interfacce di rete attive o é comunque in possesso di piú indirizzi IP, durante la fase di routing viene effettuata la scelta in base all'indirizzo destinazione e allo scope delle route disponibili. Il kernel mantiene una traccia di questa scelta in una cache, in modo tale che pacchetti successivi appartenenti allo stesso flusso vengano inviati dalla stessa interfaccia di rete e con lo stesso indirizzo IP sorgente[1]. Scelto l'indirizzo sorgente, in caso di interfacce di rete multiple, verrá scelta per l'invio del pacchetto l'interfaccia corrispondente all'indirizzo scelto.

Default gateway Quando nella tabella di routing non é esplicitamente presente una specifica route per raggiungere la destinazione, il pacchetto viene inviato al default gateway[2] indicato nella tabella di routing, spesso con 0.0.0.0/0. Solitamente un host connesso ad Internet é configurato in modo

[1] Stesso meccanismo attuato in presenza di next hop differenti descritto successivamente in "Multipath Routing"

[2] Il kernel Linux non pone restrizione sul numero di default gateway indicati nella tabella di routing, che possono quindi essere molteplici.

tale da avere una route per gli indirizzi interni alla rete locale ed un default gateway fornito dall'Internet Service Provider per essere connessi ad Internet.

4.1.2 Tabelle di routing

La tabella di routing é il cuore del processo di routing. All'interno dei sistemi Linux questa tabella viene realizzata utilizzando delle tabelle di tipo hash. Linux tipicamente possiede due differenti tabelle di routing, una per gli indirizzi locali, ossia dell'host stesso, un'altra invece con le informazioni su tutte le altre route. Mentre la prima viene solitamente popolata dal kernel, della seconda se ne occupano sia l'utente che i protocolli di routing. All'interno delle tabelle di routing le varie regole utilizzano per definire le route informazioni sulle subnet. Oltre alle tabelle di routing, nei sistemi Linux esiste un'altro tipo di struttura utilizzata nel processo di routing, una cache, in cui invece di contenere informazioni sulle subnet, vengono memorizzate le regole associate ad un singolo indirizzo IP.

Destination	Next hop	Device to use
10.0.0.0/16	10.0.1.1	*eth0*
10.0.0.0/24	10.0.0.1	*eth1*

Figura 4.2: Esempio tabelle di routing[1].

Destination	Next hop	Device to use
10.0.1.100	10.0.0.1	*eth0*
10.0.1.101	10.0.0.1	*eth0*

Figura 4.3: Esempio di routing cache[1].

Lookup Il processo di lookup di un indirizzo, attraverso il quale viene identificato il corretto destinatario del pacchetto, sia esso lo stesso host oppure un altro nodo, utilizza sia la cache che la tabella di routing, ma in modo

differente. Come primo passo della fase di lookup, viene eseguito un controllo sulla cache basato sull'esatta corrispondenza degli indirizzi; nel caso in cui l'indirizzo in questione non venga trovato, viene eseguito l'algoritmo Longest Prefix Match (LPM) sulla tabella di routing, il quale si caratterizza dallo scegliere la route piú specifica, ossia quella con la subnet di dimensione minore che combacia con l'indirizzo. Con una tabella di routing uguale all'esempio 4.2, se il processo di lookup deve individuare l'hop successivo per l'indirizzo 10.0.0.100, andrá a scegliere la seconda route. Questo perché entrambe le regole sono compatibili con l'indirizzo in questione, ma mentre la prima combacia per 16 bit su 32, la seconda combacia con 24 bit su 32 ed é quindi piú precisa. É possibile che si verifichi il caso in cui vi siano due route con la stessa precisione; in questo caso la scelta viene fatta dal Type of Service (TOS)[3]. Se anche questo non é sufficiente, viene scelta la route con la prioritá piú alta[4].

Policy di routing Di default il kernel Linux, come giá detto, utilizza due tabelle di routing; quando peró viene compilato con il supporto al policy routing, é in grado di utilizzare 255 tabelle di routing distinte e indipendenti. In questo modo é lo stesso utente che puó configurare il routing utilizzando parametri diversi, dal classico indirizzo IP di destinazione, a requisiti di Quality of Service. Quando ció avviene, prima di avviare la fase di lookup (ma dopo aver controllato la cache) il kernel deve individuare la tabella di routing da utilizzare per individuare il next hop, in base alle policies stabilite. Una volta individuata la tabella, il processo di lookup segue il suo normale sviluppo, utilizzando l'algoritmo LPM. Le policies utilizzate per la scelta della tabella di routing da utilizzare possono basarsi su molteplici parametri, utilizzabili separatamente o assieme:

[3]Il TOS é un'informazione riportata all'interno dell'header IP per specificare la prioritá del datagram in questione e il tipo di route richiesta: si puó avere una route a basso delay, ad alto throughput o altamente affidabile.

[4]Questo valore viene impostato in automatico, ma puó anche essere deciso dall'utente, per esempio tramite il comando ip route

- Indirizzo IP sorgente o destinazione;
- Device che ha ricevuto il pacchetto;
- TOS, attraverso il quale é possibile applicare regole di routing differenti in base alla tipologia di traffico (interattivo, bulk data ...)
- Fwmark: viene utilizzato per classificare il traffico soggetto a firewall. In questo caso il traffico viene classificato prima della fase di lookup della cache.

Multipath Routing Il multipath routing é una feature che permette ad un amministratore di specificare next hop differenti per la stessa destinazione nella tabella di routing. Per esempio un router potrebbe utilizzare principalmente un ISP e passare all'ISP secondario in casi eccezionali, come il malfunzionamento del primo ISP, oppure si potrebbe utilizzare il secondo next hop solamente quando viene superata una determinata soglia di bandwith richiesta ... É possibile specificare un multipath utilizzando il comando "ip route". Per esempio, il comando
"ip route add default scope global nexthop via 100.100.100.1 weight 1 nexthop via 200.200.200.1 weight 2"
imposta due differenti next hop per la stessa route, 100.100.100.1 e 200.200.200.1, specificando inoltre il livello di prioritá delle due vie, tramite il parametro weight (piú alto é il valore e maggiore é la prioritá). Una volta specificato un multipath, il kernel deve poter scegliere nella fase di lookup il next hop da utilizzare; per fare ció esistono vari algoritmi, specificati negli RFC 2991 e 2992. Questi algoritmi utilizzano il livello di prioritá (weight) assegnato dall'utente per scegliere il next hop. Di default la granularitá di questa scelta non é al singolo pacchetto, ma al numero di entries presenti nella cache di routing: quando un next hop viene selezionato, la relativa entry viene aggiunta nella cache; nel processo di lookup il kernel, prima di utilizzare le regole presenti nelle tabelle di routing, controlla la cache, quindi pacchetti consecutivi che appartengono allo stesso flusso verranno gestiti direttamente utilizzando la

cache. Per evitare ció, é comunque possibile utilizzare un'opzione che abilita il kernel ad utilizzare una granularitá al singolo pacchetto.
Di norma la scelta del next hop in caso di multipath viene fatta utilizzando un algoritmo di tipo round robin pesato, che quindi tiene in considerazione la prioritá assegnata dall'utente alle varie strade. Il kernel Linux aggiunge inoltre una componente randomica alla scelta del next hop, il modo da non rendere deterministica la sua scelta. Come detto all'inizio del paragrafo, il multipath routing coinvolge la tabella di routing. Dalla versione 2.6.12 del kernel Linux é possibile comunque sfruttare un'opzione che abilita il supporto al multipath anche per la cache di routing e che abilita l'utente-amministratore a selezionare l'algoritmo da utilizzare per la scelta del next hop.

Binding, routing e multihoming Tramite l'operazioni di binding un'applicazione puó scegliere tra gli indirizzi a disposizione del device quello da utilizzare come indirizzo sorgente per le proprie comunicazioni. Mentre questo accade molto frequentemente per le applicazioni su server, tipicamente le applicazioni client su un MN non specificano quale indirizzo usare. In questi casi é il kernel che individua l'indirizzo sorgente e quindi la relativa interfaccia di rete da utilizzare, in base alle varie regole e route presenti nella tabella di routing.
Cosa succede peró ad una comunicazione quando l'indirizzo IP sorgente prescelto non é piú in possesso del device perché l'interfaccia in questione si é associata ad una rete diversa? Le conseguenze del cambio di indirizzo dipendono dal tipo di protocollo di trasporto utilizzato, UDP o TCP:

- In caso di traffico UDP, se l'indirizzo sorgente é stato scelto dal kernel nella fase di routing, la comunicazione rimane attiva e verrá utilizzato il nuovo indirizzo IP oppure quello di un'altra interfaccia di rete. Se invece l'applicazione ha effettuato l'operazione di binding su di un'interfaccia specifica, in questo caso l'applicazione puó continuare a trasmettere, ma il kernel segnalerá il problema all'applicazione stessa (tramite valore di ritorno delle operazioni di write o send sul socket).

- Nel caso in cui invece vi sia una comunicazione TCP, indipendentemente dalle operazioni di binding, la comunicazione si blocca (il socket rimane attivo) quando l'indirizzo sorgente utilizzato non é piú in possesso del device. Occorre tener presente peró che per mantenere attiva la comunicazione non occorre che l'interfaccia utilizzata mantenga l'indirizzo, ció che risulta necessario é che il device sia sempre in possesso di quell'indirizzo, anche se associato ad un'interfaccia di rete diversa (come vedremo quando verrá analizzato HPforMSC, questo particolare risulta fontamentale nella gestione delle connessioni TCP).

4.2 Iptables

Iptables é un tool amministrativo per il filtering e il NAT di traffico IPv4[5]. Viene utilizzato per il set-up, il mantenimento e il controllo delle regole di filtering nel kernel Linux. Il funzionamento di Iptables si basa sulla definizione di catene di regole da applicare a determinati pacchetti in ingresso, in uscita dal proprio device o da inoltrare ad altre macchine. Tramite il comando user-space Iptables l'utente puó per l'appunto definire una o piú regole, specificando le caratteristiche che devono avere i pacchetti a cui applicare la regola e l'azione da eseguire, quest'ultima chiamata "target". Il kernel Linux definisce una serie di tabelle che raggruppano le varie catene di regole:

- *filter*: é la tabella di default di Iptables e contiene le catene INPUT (per i pacchetti destinati a socket presenti nella macchina), FORWARD (per pacchetti ricevuti dalla rete e che devono essere inoltrati ad altri device) e OUTPUT (per pacchetti generati localmente).

- *nat*: é la tabella consultata quando viene analizzato un pacchetto che crea una nuova connessione; contiene 3 catene, la PREROUTING (per pacchetti in ingresso ancora non oggetto del processo di routing), OUT-

[5]Per IPv6 esiste la controparte ip6tables

PUT (per pacchetti generati localmente e che non sono stati oggetto della fase di routing) e POSTROUTING (per pacchetti che stanno uscendo dal device).

- *mangle*: dal kernel 2.4.18 contiene le catene INPUT (per pacchetti in ingresso alla macchina), FORWARD (per pacchetti che devono essere inoltrati ad altri devices) e POSTROUTING (per pacchetti in uscita). Questa tabella viene utilizzata solitamente per alterare in maniera specifica i pacchetti analizzati.

- *raw*: é una tabella ad alta prioritá, maggiore di tutte le altre tabelle. Contiene le seguenti catene: PREROUTING (per pacchetti in ingresso da una qualsiasi interfaccia di rete) e OUTPUT (per pacchetti generati localmente).

Oltre a queste catene predefinite, le tabelle possono contenere ulteriori catene, definite dall'utente.

4.2.1 Processo di applicazione delle regole

Per ogni pacchetto presente in una determinata catena, viene verificato se rispetta le condizioni della prima regola presente nella catena. In caso negativo, si passa alla regola successiva. In caso affermativo invece, Iptables altera il pacchetto in base alla regola in questione (se richiesto). Alterazioni tipiche potrebbero essere il cambio di indirizzo IP sorgente o destinazione, il cambio della porta sorgente o destinazione .... Dopo l'eventuale alterazione del pacchetto, Iptables applica il target definito dalla regola, il quale andrá a specificare la successiva regola da analizzare. Questo target puó essere definito dall'utente oppure puó essere uno dei target predefiniti:

- ACCEPT: fa proseguire il pacchetto secondo il suo percorso naturale attraverso la catena in questione e le catene successive;

- DROP: elimina il pacchetto;

- QUEUE: passa il pacchetto in userspace, dove potrá essere alterato in ogni sua parte.
- RETURN: conclude il processo del pacchetto per quello che riguarda la catena di regole in questione.

4.3 Route

Route é un comando user-space di ambienti Linux attraverso il quale é possibile ottenere informazioni sulle tabelle di routing IP ed eventualmente modificarle, aggiungendo (opzione "add") o rimuovendo (opzione "del") delle regole di routing, modificando il default gateway, associando delle regole di routing ad una determinata interfaccia di rete o a tutte le interfacce a disposizione.

4.4 Ip

Ip é un comando user-space di ambienti Linux utilizzato per ottenere e modificare informazioni sul routing, sulle interfacce di rete, sulle policy di routing e sui tunnel. Piú in dettaglio, attraverso Ip é possibile manipolare le informazioni relative ai seguenti oggetti, specificabili come parametro di Ip:

- link: interfaccia di rete;
- address: indirizzo IPv4 o IPv6 di un'interfaccia di rete;
- neighbour: entry della cache ARP (Address Resolution Protocol) o NDISC (Neighbour Discovery for IPv6);
- route: entry della tabella di routing;
- rule: regole che specificano le policy di routing;
- maddress: indirizzi IP multicast;

- mroute: entry della cache di routing relative a route multicast;
- tunnel: tunnel over IP.

4.5 Hardware utilizzato

HPforMSC é stato progettato per sistemi Linux. In particolare, é stato realizzato e testato sul sistema operativo Ubuntu, con la versione del kernel 2.6.35. L'hardware utilizzato per lo sviluppo della parte client é un Cappuccino SlimPRO SP635H pci slo 64 bit, dotato di 4 interfacce di rete wireless:

- chiavetta USB 3G;
- interfaccia Wimax intel 5150; l'installazione del relativo software ha seguito i dettami del GENI Wimax Portal[6], con l'applicazione di una patch per wimax-1.5.1, dovuta all'utilizzo di hardware a 64 bit invece che a 32 bit.
- 2 interfacce WiFi 802.11 b/g, una scheda Atheros e una Ralink rt2860.

4.6 Madwifi [7]

Il progetto Madwifi é composto da un team di sviluppatori che si é posto come obiettivo quello di coordinare lo sviluppo di driver per chipset Atheros per sistemi Linux e offrire il supporto agli utenti per lo sviluppo dei driver. Utilizzando questi driver l'interfaccia WLAN appare come una normalissima interfaccia di rete, ma é possibile grazie appunto ai driver Madwifi l'utilizzo delle Wireless Extension API[8], in modo da poter configurare i vari parametri

[6] http://wimax.orbit-lab.org/wiki/WiMAX/30#a12.GENIWiMAXMobileStations

[7] Multiband Atheros Driver for Wireless Fidelity: http://madwifi-project.org/

[8] Sono API che permettono di manipolare qualsiasi device wireless in una maniera standard e uniforme. Sono composte da tre parti, l'interfaccia utente che permette di utilizzare i vari tool, una parte interna al kernel di Linux che supporta queste extension, un'interfaccia hardware implementata in ogni driver.

connessi con il device utilizzando i piú comuni tool come ifconfig, iwconfig ... Attualmente dal progetto Madwifi sono nati tre driver differenti:

- *MadWifi*: ad oggi é uno dei driver WLAN per Linux piú avanzato. É disponibile una versione stable. I driver sono open source ma dipendono da un layer software proprietario, l'Hardware Abstraction Layer (HAL). Essendo proprietario, HAL é disponibile solo in formato binario. L'ultima versione di madwifi disponibile ad oggi é la 0.9.4.

- *ath5k*: é un driver piú recente del precedente e che elimina la dipendenza del driver da HAL. Per questo motivo, negli intenti degli sviluppatori, ath5k dovrá sostituire in futuro il driver madwifi.

- *ath9k*: é il driver piú recente, sviluppato in parte da Atheros e il quale supporta tutti i chipset Atheros 802.11n.

Capitolo 5

Progettazione

HPforMSC é un'architettura per la gestione della mobilitá di un nodo mobile dotato di sistema operativo Linux, basato sull'utilizzo di un sistema di hidden proxy. Il proxy client risiede nel nodo mobile e si occupa della gestione di tutte le interfacce di rete. Il nodo mobile comunica direttamente con un solo nodo della rete Internet, il proxy server, senza che le applicazioni sul MN sappiano della presenza del proxy. Il proxy server svolge il compito di external relay, facendo da tramite tra il resto di Internet ed il MN. Tutti gli altri nodi di Internet dialogano con il proxy server, il quale fa le veci del MN.

5.1 Terminologia

Prima di addentrarsi nella descrizione di HPforMSC é meglio acquistare una certa familiaritá con la terminologia che verrá utilizzata, in modo da evitare incomprensioni e ambiguitá.

- *Payload*: con il termine payload si é indicato la parte del pacchetto di rete contenente i dati inviati dall'applicazione, escludendo quindi i vari header di trasporto, IP e MAC.

- *Stato di un'interfaccia*: un'interfaccia di rete puó essere attiva o spenta, puó essere associata ad una rete e quindi avere un indirizzo IP oppure essere attiva ma priva di indirizzo e quindi non ancora utilizzabile.

- *Interfacce virtuali*: sono interfacce di rete che non esistono fisicamente e non sono quindi associate ad alcun device fisico, ma che vengono comunque trattate dal sistema operativo come delle qualsiasi interfacce di rete. Con il termine piú generico di interfaccia di rete invece si é inteso l'interfaccia reale, associata ad un device fisico (anche se il termine "fisico" non viene specificato).

- *Catturare un pacchetto*: significa entrare in possesso del pacchetto prima che questo venga invitato dal MN, ottenere il suo contenuto e poter deciderne la sorte, ossia se inviarlo oppure no e verso che nodo inviarlo.

5.2 Hidden Proxy Client - Server

Il device mobile puó essere dotato di molteplici interfacce di rete, di tipo 3G, WiMax e WiFi 802.11, utilizzata quest'ultima sia in modalitá managed che ad-hoc. Il proxy client gestisce tutte le interfacce e i processi di associazione alle varie reti locali. Le varie applicazioni sul MN che fanno uso di una connessione ad Internet si comportano come in un normale nodo fisso, come se il proxy client non fosse presente. Quest'ultimo cattura ogni pacchetto in uscita dal nodo e, dopo aver scelto l'interfaccia di rete da utilizzare, lo dirotta verso il proxy server. Quest'ultimo puó essere un qualsiasi nodo della rete; l'unico vincolo che HPforMSC pone é che il proxy server abbia un indirizzo IP pubblico, in modo da poter essere raggiunto dal MN indipendentemente dalla sottorete in cui si trova. Il proxy server si fa carico di ricostruire il flusso di dati proveniente dal MN (identificato con un id indipendente dall'indirizzo IP) e di instaurare le connessioni richieste dal MN verso gli altri nodi della rete. Il resto di Internet conosce il proxy server ma non il proxy client. Tutti i messaggi saranno quindi inviati sempre al proxy server, il quale poi li passerá

al MN, inviandoli ad una delle sue interfacce di rete attive. Il proxy client a sua volta si occuperá di smistare tutti i messaggi provenienti dal proxy server verso le corrette applicazioni locali, le quali non si accorgeranno del dirottamento dei pacchetti in questione tramite il proxy server.

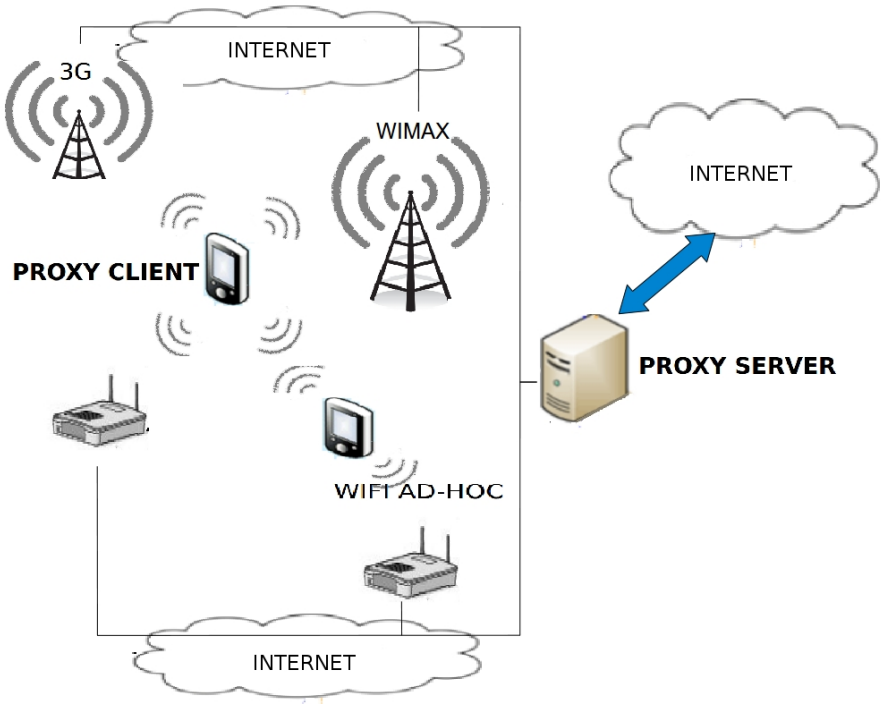

Figura 5.1: Proxy Client-Server

Grazie all'utilizzo di un external relay, mentre i nodi di Internet conosceranno il MN tramite l'indirizzo IP del proxy server, un indirizzo quindi fisso e costante, il MN da parte sua potrá comunicare con il proxy server indistintamente con una delle sue interfacce di rete attive, le quali potranno cambiare indirizzo durante la comunicazione, senza che la comunicazione stessa con il proxy server, e quindi con il resto di Internet, abbia ripercussioni. La granularitá con cui il MN puó scegliere l'interfaccia di rete da utilizzare per comunicare con il proxy server é al *pacchetto*.

5.3 Interazione proxy client - server

Le applicazioni locali nel MN generano traffico diretto verso Internet senza modificare il loro comportamento per la presenza del proxy client, presenza appunto nascosta alle applicazioni stesse. Prima peró che il traffico venga trasmesso dal MN, il proxy client lo cattura, in modo da poterlo inviare al proxy server utilizzando l'interfaccia di rete che preferisce. Prima di fare ció, estrae da ogni pacchetto catturato le informazioni necessarie per rintracciare gli originali end-point della comunicazione, ossia il CN e l'applicazione locale. Dal pacchetto viene quindi estratto il payload, al quale viene aggiunto un header contentente le informazioni sugli end-point della comunicazione. Tale messaggio viene poi inviato tramite socket UDP dal proxy client verso il server utilizzando l'interfaccia di rete prescelta. Il proxy server, da parte sua, alla ricezione di un pacchetto ne estrae l'header di HPforMSC per individuare la reale destinazione, a cui verrá passato il payload del messaggio, tramite socket (UDP o TCP). É compito del server inoltre memorizzare l'associazione tra i 2 end-point originali. In questo modo, nel caso in cui il proxy server riceva dal CN una risposta, ne estrae il payload, a cui ci aggiunge le informazioni sull'applicazione destinataria presente sul MN ed invia il messaggio al proxy client. Questo, grazie alle informazioni contenute nell'header di HPforMSC, puó passare il payload del messaggio all'applicazione locale destinataria, senza quindi che né questa né il CN si accorgano dei passaggi intermedi fatti dai loro pacchetti.

5.4 Proxy Client

Il proxy client é la parte forse piú critica e sicuramente piú sostanziale di HPforMSC. A differenza di molte altre architetture per la gestione della mobilitá, si é deciso di realizzare la parte di sw presente sul client interamente a livello applicazione, in modo tale da non dover richiedere alcuna modifica del kernel Linux, con la conseguente necessitá di ricompilare il kernel da parte dell'utente. In questo modo non risulta nemmeno necessaria una rivisitazione

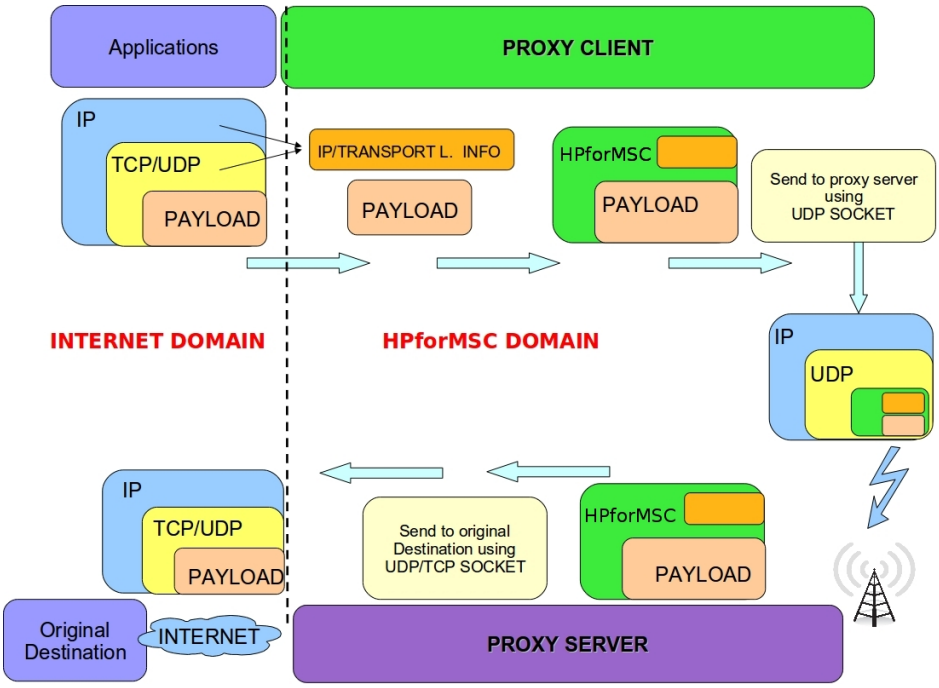

Figura 5.2: Interazione Proxy Client-Server

del sw del proxy client ad ogni nuova release del kernel Linux. Il proxy client ha il compito di mascherare all'utente e alle applicazioni ció che succede alle varie interfacce di rete, la cui gestione risulta essere completamente trasparente all'utente. Per realizzare questo importante compito, il proxy client si suddivide in vari blocchi, ognuno dei quali svolge un particolare compito:

- *Proxy Client Core*: ha il compito di catturare e gestire tutto il traffico dati in uscita alle applicazioni locali, di fare da tramite tra le applicazioni locali e il proxy server e di coordinare tutti gli altri blocchi che compongono il proxy client.

- *Interface Monitor*: ha il compito di monitorare tutte le interfacce di rete attive, di gestire tutti gli eventi collegati direttamente con le interfacce di rete, come la connessione ad una nuova rete, la disconnessione, lo spegnimento e la riaccensione dell'interfaccia ...

- *Interface Manager*: ha il compito di gestire, per ogni singola interfaccia di rete, l'invio e la ricezione di dati verso e dal proxy server.
- *TCP Manager*: ha il compito di gestire tutte le comunicazioni TCP tra le applicazioni locali ed Internet, occupandosi quindi della corretta consegna e ricezione dei messaggi, del riordino dei messaggi, della ricostruzione del flusso di dati di una comunicazione TCP ...
- *Interface Selector*: gestisce la politica di scelta dell'interfaccia da utilizzare; si occupa di definire la metrica con cui si valuta il livello di qualitá di un'interfaccia di rete e di calcolare questo valore, per ogni interfaccia attiva.
- *Application Manager*: smista il flusso di dati proveniente da Internet tramite il Proxy Server alle corrette applicazioni.

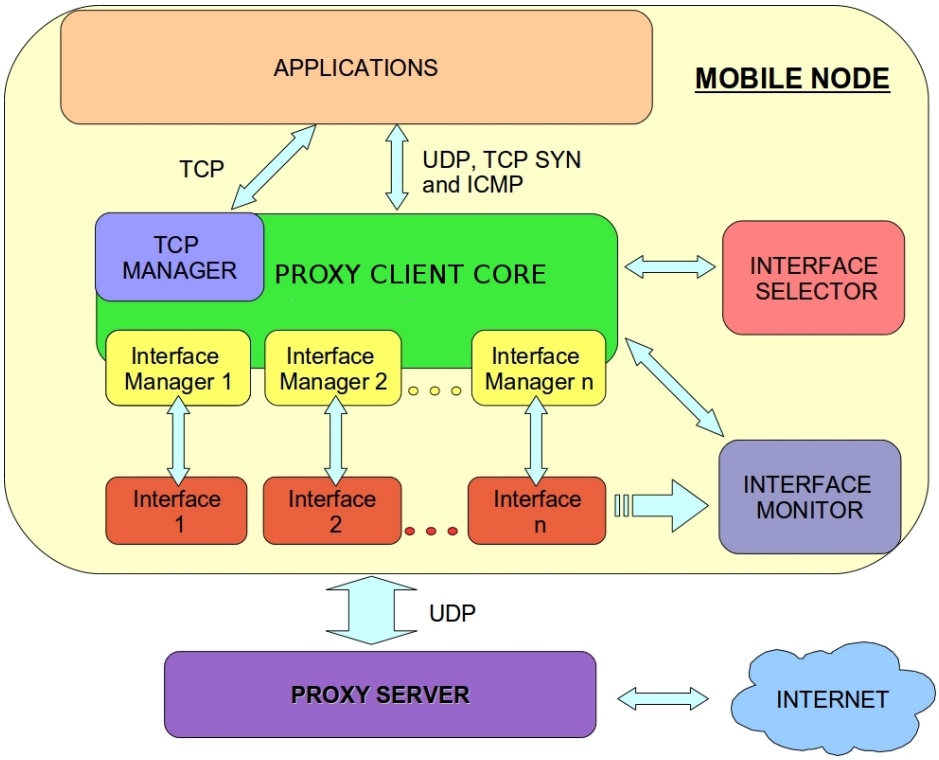

Figura 5.3: Struttura interna proxy client

5.4.1 Interface Monitor

L'interface Monitor si occupa di monitorare tutte le interfacce di rete utilizzate dal MN, della gestione della tabella di routing, di veicolare opportunamente tutto il traffico generato dalle applicazioni locali ed in uscita dal client verso il Proxy Client Core e di gestire il supporto alle connessioni TCP instaurate con indirizzi IP non piú in possesso del client.

Monitoraggio interfacce di rete Tramite un socket netlink richiede direttamente al sistema operativo di venir informato ogni qualvolta un'interfaccia di rete cambia il suo stato, ossia ogni qualvolta viene attivata o disattivata, acquisisce o perde un nuovo indirizzo IP. Per permettere al proxy client di prendere decisioni sull'interfaccia da utilizzare, l'Interface Monitor comunica al Proxy Client Core, tramite un socket, il cambiamento di indirizzo IP[1] di un'interfaccia oppure la sua attivazione o disattivazione; a sua volta, il Proxy Client Core passerá queste informazioni all'Interface Selector, il quale, come vedremo nell'omonima sezione, terrá conto di queste e di altre informazioni per valutare la miglior interfaccia di rete. Sarebbe stato possibile evitare il passaggio intermedio del Proxy Client Core e comunicare direttamente con l'Interface Selector; é stato invece deciso di far passare tutte le comunicazioni "di servizio" per il Proxy Client Core in modo da rendere gli altri blocchi dipendenti solo dal Core, rendendo quindi piú facile l'eventuale futura modifica o sostituzione di uno qualsiasi dei blocchi, ad eccezione ovviamente del Core stesso.

Deviazione del traffico Per poter gestire la mobilitá del MN senza che le applicazioni locali e l'utente se ne accorgano, occorre che il proxy client catturi tutto il traffico dati in uscita per poterlo poi manipolare e reindirizzare al proxy server. Per fare ció interamente a livello applicazione, in modo da non apportare modifiche al kernel, é stato utilizzato il tool "iptables"[2].

[1] Per cambiamento di indirizzo IP si intende sia quando l'interfaccia passa da un indirizzo ad un altro, sia quando l'interfaccia perde l'indirizzo, senza acquisirne un altro

[2] Vedi capitolo Strumenti

Tutti i pacchetti in uscita dal MN e generati dalle applicazioni locali[3] vengono passati in user space, grazie all'utilizzo del target QUEUE di iptables. In user space sará poi il Proxy Client Core a ricevere questo flusso di dati e a decidere come gestire ogni singolo pacchetto, come verrá descritto nel paragrafo dedicato al Core. La scelta di assegnare il compito di usare iptables all'Interface Monitor invece che al Core é stata presa per semplificare la gestione del proxy client e limitare i messaggi tra un blocco ed un altro, in modo inoltre da velocizzare la gestione del traffico. Infatti nell'utilizzo di iptables occorre conoscere gli indirizzi IP delle varie interfacce di rete, informazioni che l'Interface Monitor giá conosce per il suo ruolo di monitoraggio delle interfacce stesse e che quindi puó utilizzare immediatamente assieme ad iptables appena un'interfaccia acquisisce o perde un indirizzo IP, senza ulteriori ritardi dovuti alla propagazione di messaggi di controllo tra un blocco e l'altro.

Interfaccia virtuale Per il corretto funzionamento di HPforMSC, risulta necessaria la creazione di interfacce virtuali, in modo da risolvere alcuni problemi connessi con la fase di routing dei pacchetti in entrata ed in uscita al MN. Vedremo successivamente, nella sezione 5.6 i motivi per cui il proxy client necessita di queste interfacce; ció che occorre sapere quando si parla di Interface Monitor é che é proprio questo componente ad occuparsi della gestione di queste interfacce, gestione che comprende la creazione e la distruzione delle interfacce virtuali e l'aggiornamento delle tabelle di routing. Queste interfacce vengono create all'occorrenza e distrutte quando non sono piú necessarie nella fase di routing dei pacchetti. Vi é peró un'eccezione: per poter funzionare correttamente risulta infatti necessaria la presenza costante di un'interfaccia di rete virtuale; questa verrá creata dall'Interface Monitor nella fase di inizializzazione del sistema e rimarrá in vita finché HPforMSC sará in funzione.

[3]Coda OUTPUT della tabella mangle

5.4.2 Interface Manager

Esiste un Interface Manager per ogni interfaccia di rete (reale) del MN. Ogni istanza dell'Interface Manager si occupa del flusso in ingresso ed in uscita di una singola interfaccia. Per quello che riguarda il flusso in uscita, l'Interface Manager riceve i pacchetti TCP dal TCP Manager, mentre riceve tutti gli altri pacchetti (UDP e ICMP) dal Proxy Client Core. In entrambi i casi, ció che fa l'Interface Manager é semplicemente inviare questi pacchetti verso il Proxy Server, utilizzando un socket UDP. Lo stesso socket viene utilizzato anche per ricevere i pacchetti diretti al MN e provenienti dal Proxy Server. In questo caso, una volta ricevuto un intero pacchetto HPforMSC[4] l'Interface Manager si occupa di inoltrare il pacchetto al TCP Manager, in caso di traffico TCP, o al Proxy Client Core in caso contrario.

5.4.3 TCP Manager

Come detto precedentemente, la comunicazione tra proxy client e server viene fatta utilizzando UDP come protocollo di trasporto. Questo significa che, quando un'applicazione necessita una connessione TCP, i relativi pacchetti vengono inviati tramite un canale UDP, quindi senza tutti quei servizi e quelle garanzie tipiche del TCP. Occorre perció che HPforMSC si adoperi per offrire alle applicazioni servizi analoghi. Rispetto ad UDP, TCP offre:

- garanzia di consegna dei messaggi tramite meccanismi di acknoledgement;
- riordino dei pacchetti;
- possibilitá di sfruttare al massimo la capacitá del canale, tramite meccanismi di finestre di trasmissioni scorrevoli (non si attende l'ack di un messaggio per inviare i pacchetti successivi);

[4]L'eventuale frammentazione di un pacchetto HPforMSC in piú pacchetti UDP viene gestita dall'Interface Manager

- controllo della congestione sulla rete, riducendo il flusso di dati trasmessi in caso di supposta congestione quando viene perso un pacchetto e all'inizio di una comunicazione, quando ancora non si conoscono le prestazioni del canale in uso (slow start).

Attualmente gli sforzi si sono concentrati sulla garanzia di affidabilitá della consegna dei messaggi, sul riordino dei pacchetti e sull'utilizzo di finestre di trasmissione. In futuro sono peró previste delle modifiche alla gestione del TCP, in modo da aggiungere un servizio di controllo della congestione simile a quello del TCP e migliorando il meccanismo delle finestre di trasmissione, che per ora sono, a differenza del TCP, di dimensione fissa.

Quando un'applicazione locale vuole instaurare una connessione TCP verso un determinato CN, crea un socket TCP, ai cui capi vi saranno appunto l'applicazione locale e il CN. Il Proxy Client Core sostituisce al CN il TCP Manager, che di fatto diventa l'altro end-point della connessione. In questo modo il TCP Manager dialoga direttamente con le applicazioni locali tramite il socket che loro stesse hanno creato per comunicare con l'esterno, senza peró che queste si accorgano della differenza. In questo modo il TCP Manager puó estrarre il payload dai pacchetti inviati dalle applicazioni, accodarvi un header proprio per la ricostruzione del flusso di dati ed inviare attraverso l'Interface Manager appropriato il messaggio al proxy server, mediante l'interfaccia di rete migliore. Dall'altro lato, il proxy server dovrá instaurare una connessione TCP con il CN, attraverso la quale inoltrerá i messaggi ricevuti dal MN, dopo aver estratto l'header inserito dal TCP Manager. Lato ricezione, il TCP Manager riceve i pacchetti direttamente dall'Interface Manager, attraverso l'header inserito dal proxy server risale all'applicazione locale destinataria e quindi al socket corrispondente, a cui passerá il payload contenuto nel pacchetto.

Riordino dei pacchetti: Essendo la comunicazione tra client e server basata su UDP, é possibile che i dati vengano ricevuti in ordine diverso, soprattutto visto il fatto che due pacchetti consecutivi potrebbero venir inviati da interfacce di rete differenti e quindi seguire un path completamente diverso per raggiungere il server. Per poter riordinare i pacchetti, viene inserito nell'header TCP di HPforMSC un numero di sequenza. In questo modo, quando il TCP Manager (o il proxy server) riceve un pacchetto, prima di inviarlo all'applicazione locale attende di avere una porzione di dati ordinata.

Affidabilitá: Oltre all'ordine dei messaggi, utilizzando UDP viene persa anche l'affidabilitá della consegna del pacchetto. Per questo é previsto un meccanismo di acknoledgment alla ricezione di un pacchetto. Per migliorare le prestazioni della comunicazione, il messaggio di ack, oltre a contenere il numero di sequenza del pacchetto ricevuto corrispondente, contiene:

- il numero di sequenza dell'ultimo pacchetto in ordine ricevuto;
- una mappa di bit indicante l'esito degli 8 pacchetti che precedono il messaggio ricevuto e gli 8 successivi. In questo modo, da un solo ack, si hanno informazione anche sull'esito della consegna di altri 16 pacchetti, il tutto utilizzando solamente 2 byte;

Come per il TCP, é prevista una finestra di invio dei messaggi; questo significa che dopo aver inviato un pacchetto, non occorre attendere l'ack relativo prima di inviare il sucessivo, ma lo si puó inviare immediatamente, finché si rimane all'interno della finestra. Il TCP Manager si fa carico anche della gestione dei vari timeout dei messaggi inviati, in modo tale che se entro un determinato tempo dall'altro end-point non si ottiene la notifica di aver ricevuto il messaggio, tramite il corrispondente ack o tramite le informazioni presenti in un ack di un altro pacchetto (mappa di bit e numero di sequenza ultimo pacchetto in ordine ricevuto), il pacchetto viene inviato nuovamente. Se in realtá ció che é stato perso non era il pacchetto ma l'ack, l'altro endpoint riceverá un messaggio giá analizzato e quindi lo scarterá, inviando peró

nuovamente l'ack relativo al mittente. Come nel protocollo TCP, i timeout non hanno una durata costante; all'aumentare del numero di ritrasmissioni di un pacchetto il relativo timeout viene incrementato. Se, dopo un numero prefissato di tentativi, la consegna del messaggio non ha avuto successo, la comunicazione viene chiusa e viene ordinato all'altro end-point di chiudere il socket TCP corrispondente creato con il CN (proxy server) o con l'applicazione locale (TCP Manager). Ovviamente, anche per la notifica di chiusura della connessione viene garantita l'affidabilitá di consegna tramite ack.

Necessitá del TCP Manager: Il dubbio che puó sorgere analizzando il TCP Manager é relativo alla sua reale necessitá. Perché per le comunicazioni TCP invece di utilizzare un socket UDP non viene usato un socket TCP, in modo da non dover gestire ritrasmissioni e riordino dei pacchetti? Il motivo che ha portato alla progettazione del TCP Manager é il seguente: la soluzione basata su socket TCP deve prevedere una connessione TCP per ogni interfaccia di rete verso il proxy server, in modo da poter utilizzare ogni volta il socket corrispondente all'interfaccia migliore. Nel caso di perdita di un messaggio, é lo stesso sistema operativo ad occuparsi della sua ritrasmissione. Il problema é che la ritrasmissione verrá tentata sempre sulla stessa interfaccia, che, nel frattempo, potrebbe aver avuto dei problemi di connettivitá. In questo modo, prima che il proxy possa capire che il messaggio é andato perso e quindi ritrasmetterlo, eventualmente su un'altra interfaccia, verrá sprecato del tempo considerevole. Questo invece non avviene utilizzando il TCP Manager, il quale invia ogni singolo pacchetto utilizzando sempre l'interfaccia migliore, indipendentemente se il pacchetto venga trasmesso per la prima volta oppure sia una ritrasmissione. In questo modo, se un'interfaccia sta avendo dei problemi e quindi dei pacchetti vengono persi, il TCP Manager puó inviarli su un'interfaccia migliore (se esiste) dopo lo scadere di un singolo timeout TCP e non dei timeout di tutte le possibili ritrasmissioni, come nel caso basato su socket TCP.

5.4.4 Interface Selector

Come dice il nome, l'Interface Selector ha il compito di selezionare la migliore interfaccia di rete da utilizzare. Alcune architetture per la gestione della mobilitá utilizzano un'interfaccia di rete finché questa rimane associata ad un punto d'accesso della rete e solo quando questa perde connettivitá passano ad utilizzare un'interfaccia di rete alternativa. Questo fatto comporta un deterioramento nelle prestazioni durante l'handover, perché prima che un'interfaccia perda completamente connettivitá, la comunicazione subisce un aumento della perdita di pacchetti e di ritardi, dovuti quest'ultimi all'aumento di ritrasmissioni dei frame a livello 2 dello stack di rete. Per evitare questa decadenza delle prestazioni e mantenere il piú possibile costante le prestazioni, anche durante l'handover, il proxy client sceglie sempre l'interfaccia migliore[5], senza attendere la disassociazione dell'interfaccia in uso. Il compito di valutare e monitorare la qualitá delle varie interfacce e di prendere decisioni sull'interfaccia da utilizzare ricadono sull'Interface Selector, attraverso il seguente ciclo di azioni, ripetuto periodicamente:

- richiede al Proxy Client Core di inviare una serie di pacchetti UDP di controllo al proxy server su tutte le interfacce di rete attive;
- attende le risposte del proxy server;
- calcola, per ogni interfaccia di rete, il round trip time medio tra proxy client e proxy server e il numero di pacchetti persi;
- comunica al proxy client core un eventuale cambio dell'interfaccia di rete da utilizzare.

La frequenza con cui viene ripetuto questo ciclo e il numero di pacchetti di controllo da inviare per ogni interfaccia di rete sono attualmente sotto studio per poter valutare quale siano i valori migliori, in modo da rendere la

[5]La valutazione di quale interfaccia di rete sia migliore dipende da molteplici fattori, che assieme vanno a formare la "metrica per valutazione qualitá interfacce di rete", che verrá spiegata nell'omonimo paragrafo

valutazione della qualitá del canale il piú possibile accurata ma allo stesso tempo cercando di limitare al minimo le trasmissioni di messaggi di controllo, che costituiscono overhead nella comunicazione. Attualmente questi valori sono costanti e ottenuti dall'Interface Selector tramite file di configurazione. É prevista inoltre la possibilitá in futuro di rendere la frequenza di invio variabile all'interno di un prefissato intervallo di valori, a seconda di due possibili grandezze:

- la frequenza con cui variano i valori di qualitá dell'interfaccia di rete in uso: se questa é elevata significa che il suo stato é instabile, cambia rapidamente e quindi occorre aumentare la frequenza di calcolo della qualitá, in modo da rendere il proxy client piú reattivo. Al contrario, se la qualitá dell'interfaccia in uso risulta stabile, é possibile diminuire leggermente la frequenza di calcolo della qualitá.

- nel caso il device sia dotato di sensore gps, potrebbe essere efficace valutare gli spostamenti del device stesso ed aumentare la frequenza di calcolo della qualitá in caso di elevata mobilitá e al contrario renderli meno frequenti quando il device rimane pressoché fermo.

Ovviamente é possibile che, dopo aver calcolato il livello di qualitá di un'interfaccia, questa si disassoci. Nel caso l'interfaccia in questione sia quella in uso, attendere il successivo ciclo di calcolo della qualitá per effettuare il cambio di interfaccia causerebbe una grave degradazione di tutte le comunicazioni del proxy client. Per questo motivo, é previsto che l'Interface Monitor notifichi all'Interface Selector, tramite il Proxy Client Core, la disassociazione di una qualsiasi interfaccia di rete, in modo che questo possa, nel caso in cui l'interfaccia in questione sia quella in uso, comunicare al Proxy Client Core l'id della seconda migliore interfaccia di rete ancora attiva. Questa comunicazione viene effettuata immediatamente, senza quindi attendere il successivo calcolo di qualitá di tutte le altre interfacce di rete.

Metrica per valutazione qualitá interfacce di rete Il concetto di miglior interfaccia puó dipendere da molti fattori. Ad oggi, i parametri tenuti in considerazione sono:

- round trip time nella comunicazione tra proxy client e server;
- percentuale di pacchetti persi nell'ultimo intervallo di tempo analizzato;

É prevista peró in futuro la possibilitá di integrare questi parametri con un altro fattore, il costo per l'utente dell'utilizzo dell'interfaccia in termini economici. Mentre spesso il costo del WiFi non dipende dal traffico dati, questo solitamente non é valido per tecnologie come 3G e WiMax. Per questo motivo potrebbe essere lo stesso utente ad indicare all'Interface Selector le sue preferenze a riguardo.
Per ogni pacchetto di controllo inviato il proxy server deve trasmettere al client un pacchetto di risposta. Per ogni interfaccia viene calcolato il round trip time complessivo di tutti i pacchetti di controllo di cui si é ricevuto una risposta dal server. Quando l'acknowledgement é assente, al round trip time viene aggiunta una penalitá, il cui valore é ottenuto tramite file di configurazione.
Per evitare che, a causa di una temporanea interferenza sul canale, la perdita o il rallentamento di un singolo pacchetto di controllo sull'interfaccia in uso costringa il proxy client a dover cambiare interfaccia, nonostante quella in uso sia comunque la migliore, é previsto che il calcolo della qualitá delle interfacce tenga conto anche dello storico delle interfacce: il calcolo complessivo della qualitá tiene conto sia dell'esito dei messaggi di controllo inviati nella fase attuale, sia dei valori di qualitá degli ultimi "N" cicli di calcolo, con "N" definito da file di configurazione. Ovviamente, la qualitá attuale ha il peso maggiore, mentre quelle passate vedono diminuire il proprio peso man mano che si fanno temporalmente distanti dall'ultima fase di valutazione della qualitá, in modo da rendere la scelta dell'interfaccia sia indipendente dalle intereferenze molto brevi e di rilevanza esigua, sia reattiva ad interferenze

piú gravi o piú durature. La formula con cui si tiene conto dello storico di ogni interfaccia é la seguente:

$$\text{Costo complessivo}(T_i) = \sum_{k=1}^{N} \frac{Costo(T_{i-k})}{k+1}$$

Mentre il costo di un'interfaccia, ad ogni fase i, viene calcolato in questo modo:

$$Costo = \text{pkt persi}(T_i) \cdot \text{penalitá drop} + \text{round trip time pkt ricevuti}(T_i)$$

Ottenuto il costo complessivo di ogni interfaccia, il livello di qualitá viene calcolato con la seguente formula:

$$\text{Qualitá interfaccia}(T_i) = \frac{1}{\text{Costo complessivo}(T_i)}$$

Considerando il fatto che l'interfaccia in uso puó presentare un carico di lavoro notevolmente maggiore rispetto a tutte quelle interfacce di rete attive ma non utilizzate, carico che di fatto rallenta la trasmissione dei messaggi e che quindi fa lievitare il round trip time dei pacchetti di controllo, é previsto una sorta di bonus nel calcolo della qualitá per l'interfaccia in uso, che vada quindi a mitigare l'effetto negativo del carico di lavoro sull'esito della trasmissione dei messaggi di controllo.
Come giá detto, la scelta dell'interfaccia di rete da usare puó dipendere da molteplici fattori. Per rendere quindi il proxy client piú elastico ad un eventuale cambiamento della politica di scelta, é stato deciso di spostare tutto il potere decisionale sull'Interface Selector, il quale comunica con il solo Proxy Client Core, attraverso un predefinito protocollo, costituito da un insieme limitato di messaggi inviati tramite socket, in modo da rendere l'Interface Selector facilmente intercambiabile con un altro Interface Selector che implementi una politica di scelta differente. I messaggi con cui i 2 blocchi possono comunicare sono i seguenti:

- 1 messaggio da interface selector a proxy client core, attraverso il quale viene comunicato l'id della miglior interfaccia di rete da utilizzare.

- 1 messaggio dall'interface selector al proxy client core per avviare l'invio di messaggi di controllo verso il proxy server, in modo da poter valutare valori come delay e packet error rate nel canale tra client e server. L'interface selector periodicamente richiede l'invio di questi messaggi per ogni interfaccia di rete attiva. Il messaggio tra interface selector e proxy client core contiene l'id dell'interfaccia di rete su cui inviare i pacchetti di controllo, il numero e la dimensione di questi pacchetti.

- 1 messaggio da proxy client core all'interface selector, con cui il primo passa al secondo i vari messaggi inviati dal proxy server in risposta ai precedenti pacchetti di controllo inviati dal proxy client su richiesta dell'interface selector.

- 1 messaggio dal Core all'Interface Selector per comunicare che una determinata interfaccia non é piú associata ad alcuna rete.

5.5 Application Manager

Quando il Proxy Server invia dei messaggi provenienti dal CN al proxy client, il Core, dopo averli processati, li passa all'Application Manager. Quest'ultimo ha il compito di sovrascrivere l'header del protocollo di trasporto (TCP, UDP o ICMP) del pacchetto ricevuto, in modo tale da farsí che la sorgente risulti il CN e non il proxy server, eliminando quindi ogni traccia sia del proxy client che del server. Le informazioni necessarie sul CN vengono passate all'Application Manager dal Proxy Client Core, assieme al pacchetto stesso. Una volta sistemato il pacchetto, l'Application Manager lo invia all'applicazione locale corretta, tramite un socket raw[6]. Le informazioni necessarie per risalire alla corretta applicazione, ossia numero porta e indirizzo IP associati, sono anch'essi passati dal Proxy Client Core assieme al resto del messaggio.

[6]Il messaggio inviato sul socket raw contiene il payload del CN piú gli header dei protocolli network e transport

5.6 Routing nel Client

Il routing nel MN é un punto critico di HPforMSC. Alcune soluzioni prevedono di introdurre un'interfaccia virtuale e definire una sola regola nella tabella di routing che indichi come default gateway un indirizzo appartenente alla stessa subnet dell'interfaccia virtuale, cancellando tutte le altre route relative alle interfacce di rete fisiche. In questo modo nella fase di routing a tutti i pacchetti in uscita il kernel assegnerá come indirizzo sorgente l'indirizzo dell'interfaccia virtuale, perché nella stessa sottorete del default gateway. Approcci del genere sono possibili ma comportano comunque delle complicazioni nella gestione del traffico e di alcuni protocolli di trasporto come ICMP. Soprattutto peró, andando a svuotare completamente la tabella di routing, entrano in conflitto con i protocolli di routing per reti ad-hoc, in quanto vanno ad eliminare qualsiasi route individuata da questi protocolli.
Per evitare tutte queste problematiche, é stato scelto un altro approccio: viene creata un'interfaccia virtuale (permanente) e viene aggiunta una route verso un default gateway virtuale alla stessa maniera degli approcci appena descritti. HPforMSC peró non prevede l'eliminazione di tutte le altre regole presenti nella tabella di routing riguardanti le interfacce reali, ma solo dei loro default gateway. Queste informazioni non vengono perse del tutto, ma vengono riscritte nella tabella di routing, sfruttando la possibilitá offerta dal kernel Linux di definire, con una singola regola, nexthop differenti. Tutti i default gateway, compreso quello virtuale, vengono inseriti in una singola regola, la quale indica come nexthop prioritario il default gateway associato all'interfaccia virtuale[7]. In questo modo le informazioni contenute nella tabella di routing non vengono perse e il funzionamento dei vari protocolli di routing per reti ad-hoc non viene intralciato. Dopo l'azione del proxy client la parte di tabella che interessa i default gateway risulterá a questo modo (l'output di esempio é stato ottenuto tramite il comando "ip route"):
10.0.0.3 dev tunh0 proto kernel scope link src 10.0.0.2 (interfaccia virtuale

[7]Durante la fase di progettazione e testing é stata utilizzata una prioritá 100 volte maggiore

permanente)
192.168.1.0/24 dev wlan1 proto kernel scope link src 192.168.1.130
131.179.136.0/24 dev wlan2 proto kernel scope link src 131.179.136.86
default
netxhop via 10.0.0.3 dev tunh0 weight 100
netxhop via 192.168.1.254 dev wlan1 weight 1
nexthop via 131.179.136.1 dev wlan2 weight 1

Per quello che riguarda le altre route dirette verso nodi specifici diversi dal default gateway, queste risulteranno ancora presenti nella tabella di routing. In questo modo viene evitato il conflitto tra HPforMSC e l'eventuale protocollo di routing ad-hoc, il quale potrá continuare a inserire route sulla tabella di routing (il proxy si interesserá solamente della route diretta verso il default gateway).
Il fatto di assegnare maggior prioritá al gateway "virtuale" comporta che nella fase di routing venga scelto come indirizzo sorgente con una frequenza molto alta l'indirizzo associato all'interfaccia virtuale. Questo risulta essere molto importante soprattutto per la gestione delle comunicazioni TCP. Infatti, come giá descritto nella sezione riguardante il routing (capitolo Strumenti), é necessario che l'indirizzo locale che il processo di routing ha assegnato alla connessione come indirizzo sorgente rimanga sempre in possesso del MN, altrimenti la connessione TCP si blocca. Avendo maggior prioritá la route associata all'interfaccia virtuale, la maggior parte delle connessioni TCP che il MN cerca di instaurare utilizzano come indirizzo sorgente quello dell'interfaccia virtuale, che ovviamente rimane fisso, cosicché il cambio di indirizzo da parte delle interfacce fisiche non coinvolge queste connessioni. Ovviamente peró, anche se con bassa probabilitá, é possibile che il processo di routing associ ad una connessione TCP l'indirizzo di un'interfaccia fisica. Ogni volta che ció accade, l'Interface Monitor crea un'ulteriore interfaccia di rete virtuale, a cui assegnerá l'indirizzo dell'interfaccia fisica, una volta che questa lo perderá perché si é associata ad un'altra rete. In questo modo il MN rimane in possesso dell'indirizzo IP sorgente della connessione TCP, che

puó quindi rimanere attiva. Una volta che questa verrá terminata dall'applicazione proprietaria, l'Interface Monitor distruggerá l'interfaccia virtuale relativa. L'indirizzo assegnato all'interfaccia virtuale per mantenere attiva la connessione TCP é peró in possesso al MN in maniera abusiva perché non piú autorizzato dal DHCP server della vecchia rete. Perció l'Interface Monitor si occupa, dopo aver assegnato l'indirizzo all'interfaccia virtuale, di definire delle regole tramite iptables che blocchino il traffico esterno diretto verso l'indirizzo in questione[8]. In questo modo l'indirizzo avrá validitá solo all'interno del MN, senza interferire con l'esterno del mondo.
Ovviamente, come cambiano gli indirizzi IP delle interfacce reali, possono cambiare anche i relativi default gateway. Perció, ogni volta che ció accade, occorre aggiornare la tabella di routing sostituendo gli indirizzi IP delle interfacce in questione e dei relativi default gateway. Questo compito viene svolto dall'Interface Monitor, il quale, come per lo stato delle interfacce, richiede al sistema operativo di venir informato ad ogni cambiamento delle tabelle di routing, cambiamenti effettuati dai vari gestori delle interfacce di rete, come wpa_supplicant, il Network Manager di Ubuntu ... In questo modo l'Interface Monitor annulla i cambiamenti effettuati da questi gestori e riscrive le route per i default gateway secondo il criterio descritto in precedenza.

5.6.1 Proxy Client Core

Il ruolo principale del Proxy Client Core é quello di coordinatore di tutti i precedenti blocchi e di amministratore di tutto il traffico generato da applicazioni locali, diretto verso l'esterno e non gestito dal TCP Manager. Piú in dettaglio, i compiti del Proxy Client Core sono i seguenti:

- Avviare tutte le parti che costituiscono il proxy client.

[8]Per il traffico in uscita il problema non sussiste, perché é lo stesso proxy client a decidere l'indirizzo sorgente da utilizzare

- Catturare il traffico generato localmente dalle applicazioni. Come giá descritto, l'Interface Monitor, utilizzando iptables, richiede al sistema operativo di passare in user space tutto il traffico presente nella catena OUTPUT. A questo punto il proxy Client Core acquisisce tutto il traffico generato dalle applicazioni, ne estrae il payload e tutte le informazioni presenti nell'header IP e di trasporto utili per risalire al reale destinatario e alla sorgente e passa il messaggio risultante all'Interface Manager indicato dall'Interface Selector, che lo invierá al proxy server. Per il traffico TCP occorre un discorso a parte. Come giá detto, questa tipologia di pacchetti viene gestita dal TCP Manager, che si sostituisce alla reale destinazione nel socket creato dall'applicazione. Questo é possibile grazie all'intervento dell'Interface Monitor, il quale richiede al sistema operativo, tramite iptables, di dirottare tutti i pacchetti TCP di tipo syn[9] al TCP Manager, che potrá in questo modo sostituirsi al CN senza che l'applicazione se ne accorga. Quando peró iptables modifica la destinazione di un messaggio, l'informazione sul destinatario precedente viene persa e quindi il TCP Manager non conosce piú le coordinate del CN, perché i pacchetti ricevuti indicheranno come destinatario proprio il TCP Manager. Per questo motivo risulta necessaria l'azione del Proxy Client Core: prima di veder modificata la propria destinazione, il pacchetto syn viene passato in user space e quindi al Core come tutti gli altri messaggi in uscita. Il Proxy Client Core puó quindi salvare le informazioni sul reale destinatario e passarle al TCP Manager, lasciando inalterato il pacchetto e il suo percorso, che verrá poi dirottato da iptables verso il TCP Manager.

- Individuare gli eventuali pacchetti che devono essere gestiti dal sistema operativo e non dal proxy, che quindi non deve modificarne il contenuto e il percorso. Fanno parte di questa categoria per esempio le richieste dirette al DHCP server di una delle sottoreti a cui é connessa una delle

[9]I pacchetti di tipo syn vengono utilizzati nella fase iniziale di una comunicazione per notificare all'altro end-point la volontá di avviare una connessione TCP

interfacce, i messaggi di controllo dei protocolli di routing per reti ad-hoc, che ovviamente dovranno essere sempre inviati sull'interfaccia di rete ad-hoc e non dovranno passare per il proxy server. Per individuare questi pacchetti, il Proxy Client Core si basa sulla porta utilizzata[10], la quale per questi servizi é ben definita.

- Inviare, su richiesta dell'Interface Selector, i pacchetti di controllo verso il Proxy Server per definire la qualitá dei vari canali di comunicazione. In questo caso, tutte le variabili relative ai pacchetti, come dimensione, numero, interfaccia da usare, sono stabilite dall'Interface Selector. Il Proxy Client Core é un mero esecutore.

- Smistare i pacchetti in uscita dal MN verso l'interfaccia di rete migliore e quindi verso il relativo Interface Manager. Il Proxy Client Core riceve gli update sull'interfaccia di rete da usare direttamente dall'Interface Selector ed ha inoltre il compito di comunicare questi anche al TCP Manager, in modo che anche questo possa smistare i pacchetti in uscita (solo TCP) verso l'interfaccia migliore.

- Ricevere dai vari Interface Manager tutti i pacchetti provenienti dal proxy server e quindi dai vari CN, ad eccezione dei pacchetti TCP, che, come giá detto, passano direttamente dall'Interface Manager al TCP Manager. Il Proxy Client Core amministra questi messaggi ed inoltra all'Application Manager quelli che hanno come destinatario un'applicazione locale.

5.7 Proxy Server

Il proxy Server puó essere suddiviso in due parti:

- *Register Manager*: questo blocco ha il compito di effettuare la registrazione di un nuovo proxy client. Il Register Manager attende su una

[10]Con il termine porta si intende la porta definita nei vari protocolli di trasporto

porta predefinita le richieste dei nuovi MN. Quando riceve una richiesta, assegna un Client Manager al proxy client, notificando al primo l'identificativo del client e al secondo gli estremi (indirizzo IP e porta) del Client Manager, in modo che i 2 soggetti possano comunicare senza dover coinvolgere nuovamente il Register Manager.

- *Client Manager*: é il gestore di un singolo proxy client. Dal Client Manager passa tutto il traffico del suo client verso Internet e da Internet verso il client. É sul Client Manager che si concentrano tutte le funzionalitá del proxy server.

5.7.1 Client Manager

Il Client Manager riceve tutto il traffico proveniente dal proxy client presente sul MN e lo inoltra ai vari CN, prendendo il posto del MN nella comunicazione con il CN, in modo tale che, indipendentemente dall'indirizzo IP utilizzato dal MN, il CN non ne risenta perché a conoscenza solamente del proxy server. Quest'ultimo é agli occhi del CN l'altro end-point della comunicazione.

Gestione TCP Come per il proxy client, anche lato server il traffico TCP richiede una gestione particolare. Il Client Manager presente sul proxy server implementa gli stessi meccanismi del client per garantire l'ordinamento dei pacchetti e la loro consegna affidabile[11]. Per quello che riguarda invece la comunicazione TCP tra proxy server e CN, questa non necessita di gestioni particolari, perché quando il proxy client invia un pacchetto syn per notificare la richiesta di una nuova connessione TCP da parte di un'applicazione del MN, il Client Manager instaura la connessione direttamente con il CN, utilizzando un socket TCP. Perció da questo lato della comunicazione é lo stesso sistema operativo che gestisce il protocollo TCP.

[11] Vedi sottosezione 5.4.3

Associazione tra MN e CN Indipendentemente dal protocollo di trasporto utilizzato, il Client Manager deve mantenere un'associazione tra il reale mittente presente sul MN e il destinatario presente sul CN. Per rendere ció possibile, il proxy client aggiunge ad ogni pacchetto le informazioni sui reali end-point della comunicazione, costituiti dagli indirizzi IP e dalle porte utilizzate dall'applicazione sul MN e dal CN. Il Client Manager memorizza questa associazione (tramite tabella hash) cosicché puó, alla ricezione di un messaggio da parte del CN, risalire alle coordinate del destinatario reale e inserirle a sua volta nel pacchetto che invierá al proxy client, in modo che quest'ultimo possa poi utilizzare queste informazioni per indirizzare il pacchetto all'applicazione corretta.

Indirizzo IP del MN Il MN puó inviare pacchetti al server utilizzando una qualsiasi interfaccia di rete e quindi utilizzando uno qualsiasi degli indirizzi IP in suo possesso. Il Client Manager, da parte sua, non conosce lo stato delle varie interfacce, quindi, quando deve inviare un messaggio verso il MN, semplicemente invia il pacchetto utilizzando come destinatario l'ultimo indirizzo IP utilizzato dal MN per comunicare con il server. Questa strategia, seppur semplice, é abbastanza efficace, perché presumibilmente l'indirizzo in questione sará quello dell'interfaccia di rete migliore, visto la politica di scelta dell'interfaccia del proxy client (vedi sezione 5.4.4). É tuttavia possibile con poco sforzo implementare una strategia alternativa, in cui per esempio é il MN stesso ad indicare al proxy server quale indirizzo IP utilizzare come destinatario.

Messaggi di controllo per la qualitá del canale Come descritto nella sezione 5.4.4 relativa all'Interface Selector il proxy client periodicamente invia dei messaggi di controllo diretti al server per calcolare il rtt e la percentuale di pacchetti persi per ogni interfaccia. Quando il Client Manager riceve uno di questi pacchetti, invia immediatamente un messaggio di risposta al MN diretto sull'interfaccia utilizzata per inviare la richiesta.

5.7.2 Server multipli

Il vincolo che esiste tra MN e proxy server non é immutabile. Infatti, in base alla posizione del MN, é possibile, dopo essersi associati ad un proxy server, instaurare nuove comunicazioni con un proxy server piú vicino (o con un server con meno carico di lavoro). In questo caso le vecchie comunicazioni saranno gestite sempre dal primo server, mentre le successive potranno essere gestite dal server piú vicino, in modo da diminuire la latenza tra MN e CN. Questo significa anche che, se il proxy server viene gestito tramite una farm di macchine oppure tramite cloud, é possibile aggiungere risorse (ulteriori server) all'aumentare dei MN o delle loro richieste anche a runtime, senza coinvolgere gli utenti e le prestazioni delle loro comunicazioni.

5.8 Header HPforMSC

Quando il proxy client e il server si inseriscono nella comunicazione tra le applicazioni nel MN e nel CN, devono in qualche modo mantenere le informazioni sugli originali end-point della comunicazione, in modo da poter risalire, dato un qualsiasi pacchetto scambiato tra i 2 proxy, a sorgente e destinazione originale. Per fare ció, in ogni pacchetto i proxy client (Proxy Client Core e TCP Manager) e server (Client Manager) inseriscono, tra payload e l'header del protocollo di trasporto, un ulteriore header definito da HPforMSC. In questo header vengono memorizzati:

- identificativo proxy mittente, indipendente dall'indirizzo IP, in modo da non dover essere influenzato dai vari handover;
- indirizzo IP della destinazione originale;
- numero porta protocollo di trasporto della destinazione originale;
- indirizzo IP della sorgente originale;
- numero porta protocollo di trasporto della sorgente originale;

- tipo di protocollo di trasporto originale (UDP, TCP o ICMP);
- dimensione payload

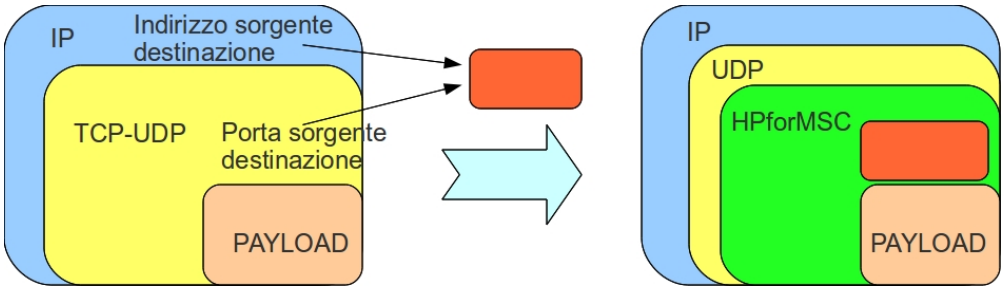

Figura 5.4: Struttura header HPCforMSC

Traffico TCP Come giá detto piú volte, inviare pacchetti TCP attraverso una comunicazione UDP richiede sforzi aggiuntivi per mantenere le caratteristiche di affidabilitá della consegna e ordine dei messaggi del TCP. Per rendere possibile tutto ció HPforMSC prevede un ulteriore header dedicato ai soli pacchetti TCP, inserito tra l'header descritto in precedenza e il payload del messaggio. Questo header HPforMSC-TCP contiene:

- tipologia pacchetto; sono previsti gli stessi tipi previsti da TCP, ma attualmente sono utilizzati (in maniera non esclusiva) solo le tipologie syn, ack, data, err;
- sequence number, in modo da garantire il riordino;
- eventualmente, nel caso di ack, i campi definiti nella sottosezione 5.4.3, paragrafo relativo all'affidabilitá.

Capitolo 6

Note Implementative

6.1 Interface Monitor

Per quello che riguarda l'implementazione dell'Interface Monitor, i punti piú salienti ed importanti sono sicuramente il recupero delle informazioni sulle varie interfacce di rete e la gestione della tabella di routing, in particolare l'utilizzo di default gateway multipli.

6.1.1 Informazioni sulle interfacce di rete

L'interface Monitor ottiene dal sistema operativo due tipi di dati, informazioni sullo stato delle interfacce di rete ed informazioni sulla tabella di routing. In entrambi i casi il mezzo utilizzato é lo stesso, un socket NETLINK. Questa tipologia di socket é un sistema IPC (Inter-Process Communication) utilizzato tipicamente per permettere la comunicazione asincrona full-duplex tra kernel space e user space [1]. É possibile ottenere le informazioni necessarie per l'Interface Monitor tramite socket NETLINK specificando le seguenti opzioni:

- RTMGRP_LINK: cambiamento dello stato[2] di un'interfaccia di rete;

[1] Altri meccanismi per la comunicazione user-kernel ma non basati su socket sono per esempio system call e file system virtuali come proc, sysfs

[2] L'interfaccia viene creata, distrutta, attivata, disattivata

- RTMGRP_IPV4_IFADDR: acquisizione e perdita di un indirizzo IPv4 da parte di un'interfaccia di rete;
- RTMGRP_IPV4_ROUTE: cambiamento di una route nella tabella di routing;
- RTMGRP_NOTIFY: cambiamento di un indirizzo nella tabella di routing;

In questo modo il sistema operativo, ad ogni cambiamento di uno dei precedenti oggetti, invia un messaggio sul socket NETLINK su cui l'Interface Monitor é in ascolto[3]. Il passaggio di informazioni consiste in uno stream di byte costituito da uno o piú header di tipo nlmsghdr, con i relativi payload. Questo stream di dati é accessibile tramite le macro NLMSG_*[4].

6.1.2 Creazione interfaccia virtuale

Ogni interfaccia virtuale creata dall'Interface Monitor é di tipo TUN, la quale quindi simula un device di rete a livello network dello stack ISO/OSI ed opera con pacchetti di livello 3, quali per esempio i pacchetti IP. Un'interfaccia di tipo TUN puó essere vista come un device ethernet oppure point-to-point, il quale, invece di ricevere pacchetti da un'interfaccia fisica, li riceve da processi in user space ed invece di trasmettere pacchetti verso un media fisico li trasmette verso un processo in user space.
Per creare un'interfaccia virtuale occorre aprire */dev/net/tun* ed eseguire l'operazione di ioctl corrispondente per la registrazione di un device di rete sul kernel, selezionando inoltre il nome dell'interfaccia, solitamente del tipo "tunXX" con al posto di "XX" un numero sequenziale (1 nell'esempio successivo). Una volta completata con successo l'operazione di ioctl, questa nuova

[3]Si tratta di una comunicazione multicast tra kernel e user space, questo significa che contemporaneamente all'Interface Monitor potrebbero esserci altri processi in ascolto delle stesse informazioni

[4]Per esempio alcune macro utilizzate sono NLMSG_OK, NLMSG_DATA, NLMSG_NEXT

interfaccia virtuale avrá vita finché il processo che l'ha creata non la chiuderá oppure finché lui stesso rimarrá in vita.

```
struct ifreq ifr;
fd = open(/dev/net/tun, O_RDWR));
ifr.ifr_flags = IFF_TUN;
strncpy(ifr.ifr_name, tun1, IFNAMSIZ);
ioctl(fd, TUNSETIFF, (void *) &ifr);
```

6.2 Catturare i pacchetti con iptables

Fulcro centrale nel funzionamento del proxy client é la cattura di tutti i pacchetti in uscita dal MN e generati dalle applicazioni locali. Come giá descritto nell'ananlisi del monitor in 5.4.1, il traffico in uscita viene catturato utilizzando una funzionalitá di iptables che permette di deviare i pacchetti prima che vengano trasmessi, per passarli ai processi che ne fanno richiesta in user space. Per ottenere ció, vi sono due operazioni da fare, la prima da parte dell'Interface Monitor, che segnala al sistema operativo quali pacchetti deve passare in user space, la seconda invece per ricevere e manipolare i suddetti pacchetti.
Tramite il seguente comando
iptables -t mangle -A OUTPUT -d ! 10.0.0.2 -m iprange ! –dst-range 127.0.0.0-127.0.0.255 ! –src-range 127.0.0.0-127.0.0.255 -j QUEUE
é stato deviato in user space (QUEUE) tutto il traffico generato da un applicazione locale e non diretto né all'interfaccia virtuale permanente creata dall'Interface Monitor (*-d ! 10.0.0.2*), né diretto ad un indirizzo locale (*–dst-range 127.0.0.0-127.0.0.255 !*).
Richiesta la deviazione dei pacchetti, questi sono stati catturati dal Proxy Client Core nel seguente modo:
Viene creato un gestore per i pacchetti passati in user space:

*struct ipq_handle *h*

int h = ipq_create_handle(0, PF_INET)

Viene richiesto al sistema operativo di copiare in user space l'intero pacchetto e non solo le informazioni ausiliarie ad esso associate:

int rval = ipq_set_mode(h, IPQ_COPY_PACKET, BUFSIZE)

Ci si mette in attesa che il sistema operativo passi un pacchetto, come se si fosse in attesa su di un socket qualsiasi[5]. Ad ogni messaggio ricevuto, si controlla la tipologia tramite *ipq_message_type* in modo da escludere gli eventuali messaggi di errore e analizzare solo i pacchetti di rete (tipologia IPQM_PACKET).
Dal messaggio ricevuto é possibile estrarre una struttura di tipo *ipq_packet_msg_t* contenente il pacchetto e alcune informazioni ausiliarie, tramite

*ipq_packet_msg_t *m = ipq_get_packet(packet)*

Da questa struttura é poi possibile estrarre il contenuto del pacchetto, comprendente i dati generati dall'applicazione e gli header di trasporto (TCP, UDP o ICMP) e IPv4. Una volta ottenuto il pacchetto, é possibile, tramite *ipq_set_verdict*, deciderne la sorte: il pacchetto puó proseguire il suo cammino normale oppure puó essere eliminato. Occorre tener presente che, mentre la ipq_set_verdict agisce sul pacchetto originale, i dati ottenuti in user space sono solo una copia. Questo significa che eventuali modifiche effettuate in user space in realtá non coinvolgono il messaggio originale e che verrá inviato, ma solamente una sua copia. Per superare questo scoglio, ottenuto il payload e le informazioni sui due estremi della comunicazione, il pacchetto originale viene eliminato[6] con ipq_set_verdict e ne viene creato un altro contenente lo stesso payload, piú le informazioni contenute nell'header HPforMSC.

6.2.1 Traffico TCP

Per la gestione del traffico TCP, come giá descritto in precedenza, viene seguito un procedimento separato. Mentre il TCP Manager rimane in attesa

[5]In questo caso specifico, é stata utilizzata una select

[6]Per i pacchetti TCP vedere la sottosezione seguente

su un socket associato ad una porta locale prestabilita, l'Interface Manager dirotta i pacchetti TCP di tipo syn sulla porta del TCP Manager, tramite: *iptables -t nat -A OUTPUT -p tcp –syn -m iprange ! –dst-range 127.0.0.0-127.0.0.255 -j REDIRECT –to-port TCP_MANAGER_PORT*

Il TCP Manager, da parte sua, esegue un accept sul socket sopracitato. Questa syscall estrae la prima richiesta di connessione (che coincide con il pacchetto syn del TCP) pendente sul socket, crea un nuovo socket a cui associa la richiesta e ne restituisce l'identificativo, mantenendo invariato lo stato del socket originale, che sará ancora in attesa di altre richieste. In questo modo il TCP Manager ottiene un socket per ogni richiesta di connessione TCP effettuata dalle applicazioni locali e diretta all'esterno del MN, socket attraverso il quale puó quindi dialogare con le applicazioni sostituendosi al CN, senza che l'applicazione stessa se ne accorga. Come descritto nella sezione dedicata alla progettazione, le informazioni necessarie per conoscere gli estremi del CN (indirizzo IP e porta) vengono passate al TCP Manager dal Proxy Client Core, tramite socket.

Capitolo 7

Valutazione

La valutazione di HPforMSC é stata effettuata seguendo vari criteri e con molteplici obiettivi. Si é voluto dimostrare:

- l'effettivo funzionamento del cambio di interfaccia usata senza perdita di performance;
- le buone prestazioni durante l'handover quando la qualitá dell'interfaccia di rete utilizzata decade, valutando in questo modo anche la bontá e i difetti della politica di scelta dell'interfaccia da usare adottata dall'Interface Selector;
- il basso peso nelle performance dovuto alla fase di elaborazione dei proxy client e server;
- l'assenza di percezione del cambio di interfaccia di rete utilizzata da parte dell'utente.

Per ottenere ció sono stati effettuati test differenti, che verranno qui di seguito presentati e a cui seguirá una valutazione globale di HPforMSC per poter fare un confronto con lo stato dell'arte descritto nell'omonimo capitolo 2.

7.1 Test effettuati

É possibile suddividere la valutazione delle prestazioni di HPforMSC in due parti separate, la prima si concentra sulle tecniche adoperate per effettuare un handover, la seconda invece esamina la decisione su quando cambiare interfaccia di rete, decisione che come giá descritto si concentra completamente all'interno dell'Interface Selector. Per questo motivo i test effettuati sono stati separati in due tronconi distinti, il primo necessario per verificare che il processo di handover non comporti alcun abbassamento delle prestazioni, indipendentemente dalla politica di scelta dell'interfaccia, il secondo invece per verificare l'efficacia dell'Interface Selector nel prendere la giusta decisione sull'interfaccia di rete da utilizzare.

7.1.1 Assenza di perdita di performance durante l'handover

Questa sottosezione descrive la prima parte di test, in cui si é voluto analizzare le prestazioni di comunicazioni TCP e UDP durante un handover, senza l'influenza dell'Interface Selector, per verificare che il passaggio da un'interfaccia all'altra possa essere fatto in qualsiasi momento senza ripercussioni sulle comunicazioni. Per effettuare questo test quindi si é utilizzato un Interface Selector sostitutivo, il quale implementa una politica di scelta di tipo round robin: non vi é una valutazione della qualitá del canale, ma semplicemente viene scelta a turno ogni interfaccia per un intervallo di tempo prestabilito (10 secondi).

Traffico TCP Durante il primo test é stato effettuato un download di un video con risoluzione pari a 360p (480x360) dal sito www.youtube.com. I risultati mostrati nel grafico 7.1 mostrano la quantitá di dati ricevuti dallo stream video durante i vari handover forzati tra le 4 interfacce di rete disponibili. Per quello che riguarda l'ad-hoc, il MN comunicava con un nodo

della rete ad-hoc, il quale a sua volta era direttamente connesso ad una rete ethernet e svolgeva il compito di gateway verso Internet. É da evidenziare il fatto che per scaricare il filmato é stato utilizzato il programma VLC, caratterizzato dall'aver una bufferizzazione dei dati ridotta rispetto ai piú comuni browser, bufferizzazione che altrimenti sarebbe andata ad inficiare i risultati del test.

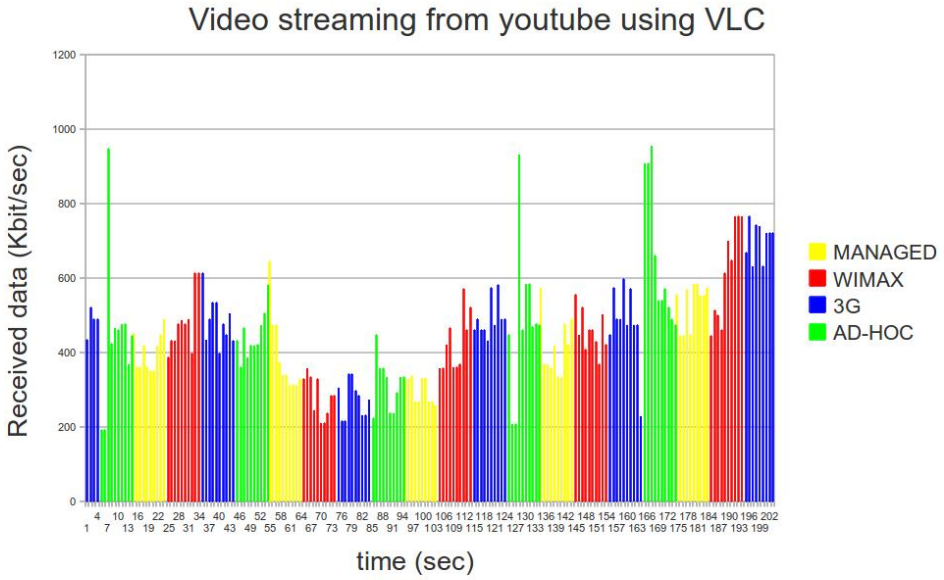

Figura 7.1: Throughput download video da Youtube, con scelta interfaccia da usare di tipo round-robin. Interfacce di tipo 3G, WiMax, Wifi modalitá managed e ad-hoc.

Discussione risultati Grazie a questo test é possibile verificare che effettivamente i cambiamenti di interfaccia effettuati da HPforMSC durante una connessione TCP non incidono in maniera evidente sul throughput della comunicazione stessa. Questo significa che se all'istante t é in uso un'inter-

faccia mentre all'istante *t+1* é in uso un'interfaccia diversa, questo non va ad incidere sul traffico dati[1].

Traffico UDP Il secondo test ha invece coinvolto il traffico UDP e si é focalizzato sull'analisi dei ritardi dei vari pacchetti inviati durante un handover. Per fare ció il MN ha inviato un pacchetto UDP di 4byte verso il proxy server ogni 100 ms. Lato server, sono stati registrati gli istanti in cui ogni pacchetto é stato ricevuto, in modo da poter verificare la presenza di dilatazioni eccessive tra la ricezione di un pacchetto e il successivo durante un handover. É da evitare infatti il caso in cui nel passaggio da un'interfaccia all'altra vi siano dilatazioni significative della latenza, non strettamente collegate con le differenti latenze delle varie tecnologie ma dovute all'intervento del proxy. Per esempio, nel caso d'uso di una comunicazione VoIP, l'eventuale burst di pacchetti arrivati con eccessivo ritardo durante l'handover andrebbe a costituire un degradamento significativo della qualitá della comunicazione stessa[2].

Discussione risultati Prendendo come punto di riferimento i primi pacchetti ricevuti dal server, il grafico 7.2 mostra quanto la ricezione dei successivi pacchetti si discosta dal valore teorico: i pacchetti vengono inviati ogni 100ms, quindi dalla ricezione dei pacchetti iniziali inviati per stabilizzare il conteggio e l'ennesimo pacchetto devono passare n per 100 ms. Tenendo conto dei possibili ritardi delle applicazioni nella rilevazione del tempo e dell'errore (minimo) introdotto dall'utilizzo dei pacchetti iniziali come punto di riferimento, dal grafico 7.2 é possibile notare che, ad eccezione del 3G, i passaggi da un'interfaccia all'altra non comportano una dilatazione nella ricezione dei pacchetti, ovviamente escludendo i ritardi strettamente connes-

[1] Ovviamente ció rimane vero entro i limiti fisici della tecnologia in uso, sia in termini di throughput che di latenza.

[2] Da specifiche ITU-T G.114 in ambito VoIP, si considera buona una comunicazione caratterizzata da una latenza massima pari a 150 ms e una percentuale di pacchetti persi inferiore al 10%.

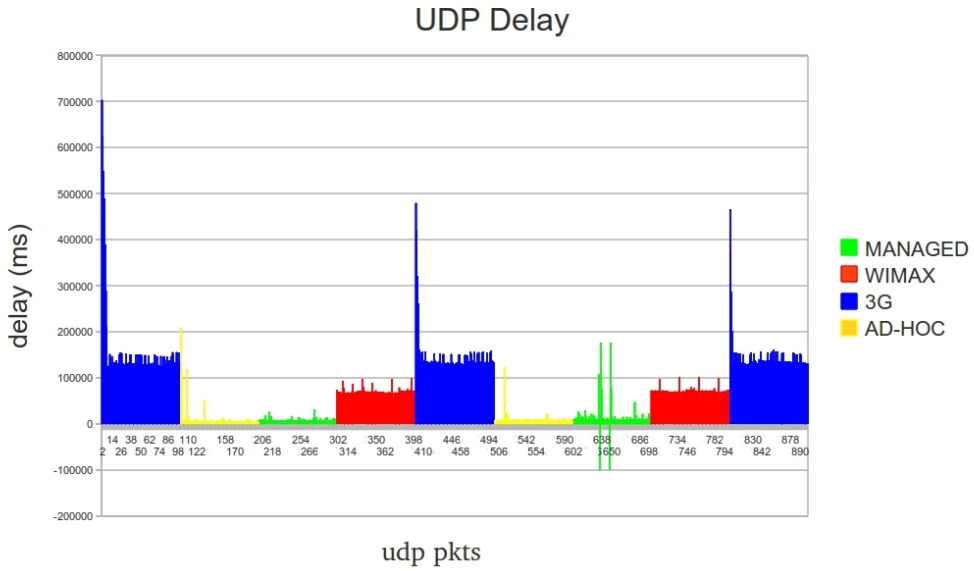

Figura 7.2: Scostamento tra l'istante teorico di ricezione dei pacchetti UDP e l'istante di effettiva ricezione, con scelta interfaccia da usare di tipo round-robin. Interfacce di tipo 3G, WiMax, Wifi modalitá managed e ad-hoc.

si con la tecnologia in uso. L'eccessivo ritardo dei pacchetti inviati tramite 3G dopo un handover (si tratta solitamente di 1-2 pacchetti UDP) é probabilmente dovuto a fattori esterni al proxy, come il calcolo della route per arrivare al proxy server da parte dei nodi del provider 3G.
I valori negativi (-100ms) presenti nel grafico rappresentano i pacchetti non ricevuti dal server (ogni pacchetto inviato dal MN conteneva nel payload un numero di sequenza, cosicché lato server era possibile individuare i pacchetti mancanti ed evitare problemi nel calcolo dei ritardi dovuti a possibili ricezione di pacchetti fuori ordine).

7.1.2 Politica di scelta dell'interfaccia

Ovviamente per valutare nella sua interezza le prestazioni di HPforMSC, occorre analizzare la bontá della politica di scelta dell'interfaccia da utilizzare, ossia dell'Interface Selector, punto forse piú critico ma anche facilmente modificabile, grazie alla scelta di spostare tutto il potere decisionale del proxy client in un oggetto ben definito e delimitato. Per analizzare il comportamento di HPforMSC é stato effettuato un test con le seguenti modalitá:

- Il MN era situato in un'automobile che si muoveva attorno al blocco riportato in figura 7.3
- Con le stesse modalitá del test precedente riguardante il traffico TCP, il MN durante il test ha scaricato un filmato da Youtube.com tramite VLC.
- L'interface Selector calcolava la qualitá del canale ogni 2 secondi.
- Ad ogni ciclo, l'Interface Selector richiedeva l'invio di 2 pacchetti di controllo per ogni interfaccia di rete attiva.

Discussione risultati Il grafico 7.4 mostra risultati prevalentemente positivi, anche se con qualche eccezione. In alcuni casi la politica di scelta dell'Interface Selector é efficace, anticipando il cambio dell'interfaccia prima che le prestazioni di quella in uso calino drasticamente. Alcune volte invece si riscontrano dei cali nel throughput. I motivi principali che causano questi cali sono i seguenti:

- Quando il calo precede un handover, significa che l'Interface Selector non ha agito tempestivamente ordinando un cambio di interfaccia. Questo dipende principalmente dalla frequenza di update della qualitá del canale, posta durante i test ad 1 update ogni 2 secondi, ma ancora sotto analisi per individuare il valore corretto, come detto anche in

Figura 7.3: Percorso veicolo durante i test e copertura WiFi, 3G e WiMax

precedenza. Come vedremo nel capitolo dedicato agli sviluppi futuri, é previsto l'utilizzo di una tecnica cross-layer per una individuazione piú tempestiva della degradazione del canale.

- Quando il calo si trova temporalmente distante da un handover, i motivi sono principalmente 2: é possibile che alcuni pacchetti di controllo sulle interfacce non in uso siano andati persi e quindi l'Interface Selector abbia privilegiato un canale piú sicuro anche se con latenza maggiore. Un altra causa risiede invece nella copertura dell'area in cui sono stati svolti i test; infatti erano presenti 2 zone intorno al blocco in cui nessuna delle 3 tecnologie in dotazione aveva una buona copertura, il che ha ovviamente comportato una degradazione delle prestazioni a cui HPforMSC non poteva porre rimedio.

Il basso utilizzo della tecnologia WiFi osservabile nel grafico 7.4 trova la principale motivazione nella bassa copertura da parte dell'access point WiFi, copertura che di fatto interessava un solo lato del blocco attorno al quale

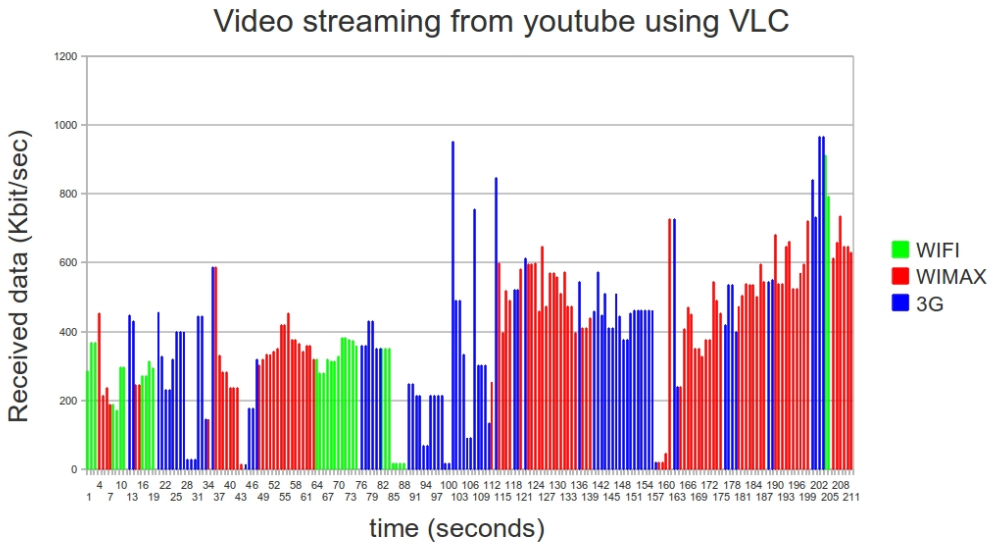

Figura 7.4: Throughput download video da Youtube, con utilizzo best-interface. Interfacce di tipo 3G, WiMax e Wifi modalitá managed.

girava l'automobile. Questo ha comportato che ad ogni giro il MN é riuscito ad utilizzare l'interfaccia WiFi solamente per un breve tratto, ancor piú breve del lato coperto dall'AP, a causa dei tempi richiesti dalla tecnologia per l'associazione dell'interfaccia all'AP stesso.

7.1.3 Carico di lavoro di HPforMSC

Mostrato come durante un handover, salvo qualche eccezione, le prestazioni del sistema non subiscono un peggioramento significativo, occorre verificare che le tecniche messe in atto da HPforMSC per rendere tutto ció possibile non vadano ad incidere negativamente sulle prestazioni del sistema quando invece non occorre alcun cambiamento di interfaccia. Perció occorre un confronto delle prestazioni con e senza HPforMSC nell'utilizzo di una de-

terminata interfaccia di rete. Per effettuare ció é stato scelto di calcolare i valori di throughput di una comunicazione UDP tra MN e un CN posto nella stessa sottorete del proxy server. Questa particolare locazione del CN é stata scelta in modo tale da valutare gli effetti del solo carico di lavoro aggiunto dal proxy client e dal server, senza aggiungere anche gli eventuali ritardi dovuti all'utilizzo di un relay esterno che tipicamente richiede una deviazione della route. Ovviamente questo ulteriore ritardo é da tener bene presente nella valutazione complessiva di HPforMSC, ma in questo preciso caso il test in questione ha il solo obiettivo di verificare se il lavoro di smistamento dei pacchetti da parte del proxy comporta una degradazione delle prestazioni.
Per effettuare il test é stato scelto di utilizzare un'applicazione realizzata adhoc a questo scopo la quale invia con rate costante pacchetti UDP ad un CN. L'applicazione che riceverá questo traffico ha il compito di registrare le relative statistiche sulla comunicazione. Il MN invia per 60 secondi pacchetti di dimensione massima prefissata e con un rate costante. Per ogni esperimento sono stati effettuati 10 test.

Risultati Qui di seguito vengono riportati i valori medi di throughput della comunicazione in presenza e in assenza di HPforMSC, utilizzando interfaccia WiFi in modalitá managed:

- Un pacchetto da 1Kbyte ogni 15 ms:
  - *Presenza di HPforMSC*:
    - Byte trasmessi: 65.5 KBytes/sec
    - Byte ricevuti: 56 KBytes/sec
  - *Assenza di HPforMSC*:
    - Byte trasmessi: 65.5 KBytes/sec
    - Byte ricevuti: 57 KBytes/sec
- Un pacchetto da 100 byte ogni 10 ms
  - *Presenza di HPforMSC*:

    - Byte trasmessi: 9.8 KBytes/sec
    - Byte ricevuti: 9 KBytes/sec
- *Assenza di HPforMSC*:
    - Byte trasmessi: 9.8 KBytes/sec
    - Byte ricevuti: 8.9 KBytes/sec

Come si puó osservare, i valori di throughput in presenza e in assenza di HPforMSC sono pressoché identici. Questo significa che la fase di preprocessing effettuata da HPforMSC sia nel MN che nel server non va ad inficiare le prestazioni del sistema.

Distanza tra proxy server e MN Come giá evidenziato, l'inserimento di un punto intermedio tra MN e CN, comporta un aumento della latenza nella comunicazione tra MN e CN. Questa latenza é difficilmente calcolabile in maniera esatta a priori, proprio perché dipende dalla posizione del server rispetto alla route che collega MN e CN. Maggiore é la distanza tra la route originale e il server, maggiore sará il delay. Un aspetto che peró tende a favore di HPforMSC consiste nel fatto che, come ripetuto piú volte, lo scenario comune delle reti attuali prevede la frequente presenza di sistemi di NAT o di FW, i quali non permettono il return routability e quindi la comunicazione diretta tra due nodi, richiedendo perció, indipendentemente dall'architettura di gestione di mobilitá adottata, un relay esterno che faccia da tramite tra MN e CN, proprio come il proxy server fa in HPforMSC.

7.2 Handover trasparenti all'utente

Per mostrare visivamente la completa trasparenza della gestione delle interfacce e mancanza di percezione del cambiamento di interfaccia da parte dell'utente, sono stati filmati dei possibili casi d'uso di HPforMSC, un download tramite browser (Firefox) di un video su Youtube.com (traffico TCP) ed una videochiamata con chat inclusa utilizzando Skype (traffico TCP e

UDP)[3]. In queste prove non vengono mostrati dati specifici, ma viene solamente mostrato un caso reale di utilizzo di HPforMSC. In questo documento, non potendo includere per ovvie ragioni un video, vengono riportati i link alle due prove:

- Youtube: https://docs.google.com/open?id=0By-amkwLgYRNY mM0NzI3M2YtZTFhNS00NGZiLTkwMGItYzE0ZWMzZTk3OTY1
- Skype: https://docs.google.com/open?id=0By-amkwLgYRNN jkwNTIxZjEtNTU0MC00YmI3LTkzZDctMDdjOTkzNDNiN2Zk

La versione dell'Interface Selector in uso in queste prove era, come nei primi test descritti, quella semplificata di tipo round robin.

7.3 HPforMSC a confronto con lo stato dell'arte

Nel capitolo Stato dell'arte sono state valutate le varie architetture per la gestione della mobilitá utilizzando le tabelle riassuntive 2.1, 2.2 e 2.3. Riproponendo lo stesso approccio, qui di seguito viene riportata una valutazione simile di HPforMSC.

Requisiti

- *IPv6*: NO
- *MIPv6*: NO
- *IPSec*: NO

[3]Valori minimi di banda richiesti da Skype sono 128kbps, mentre il valore tipico consigliato é pari a 300kbps, sia in download che in upload. Fonte: https://support.skype.com/en/faq/FA1417/How-much-bandwidth-does-Skype-need.

Modifiche necessarie

- *Protocol stack nel MN*: NO
- *Protocol stack nel CN*: NO
- *Access network*: NO
- *Border Gateway*: NO
- *External Relay*: SI
- *Applicazione nel MN*: NO
- *Applicazione nel CN*: NO

Performance

- *Supporto a SIP/RTP*: SI
- *Supporto a TCP*: SI
- *Supporto a UDP*: SI
- *Supporto a applicazioni legacy*: SI
- *Identificazione sender*: SI
- *NIC multipli simultanei*: SI
- *Granularitá*: pacchetto
- *No limitazione FW e NAT*: SI
- *Intervallo di indisponibilitá durante un handover*: basso
- *Intervallo di continuitá*: alto
- *Latenza end-to-end*(caso ottimo): medio, alto se il server é molto lontano dal MN
- *Latenza end-to-end*(caso comune): medio, alto se il server é molto lontano dal MN

Capitolo 8

Sviluppi futuri

Gli aspetti su cui é possibile lavorare per migliorare HPforMSC sono molteplici. In questo capitolo, oltre a vedere alcune di questi elementi, verranno descritti anche alcuni possibili scenari d'utilizzo di HPforMSC che si discostano dal caso d'uso principale ed originale, la gestione della mobilitá per tutte le comunicazioni attive sul MN.

8.1 Politica di scelta dell'interfaccia

La scelta dell'interfaccia da utilizzare é una fase molto critica e da cui dipendono fortemente le prestazioni di HPforMSC. Su questo lato gli sforzi futuri si potrebbero concentrare principalmente su due aspetti:

Raffinazione calcolo qualitá : Come giá descritto in fase di progettazione, é prevista la possibilitá di rendere variabile la frequenza degli aggiornamenti della qualitá delle varie interfacce e il numero di pacchetti di controllo inviati ogni volta. La variazione dipenderebbe dagli spostamenti del MN (tramite sensore GPS, attualmente in uso in moltissimi device mobili) e dalla frequenza con cui varia la qualitá del canale in uso.
É previsto e necessario inoltre un ulteriore e piú approfondito studio per valutare la bontá della formula per il calcolo della qualitá, le modalitá con cui

viene tenuto in considerazione lo storico di ogni interfaccia e la "penalitá" introdotta in caso di mancata ricezione del pacchetto.

Integrazione con TED Il Transmission Error Detector (TED)[25] é un componente dell'architettura ABPS (Always Best Packet Switching) a cui ho in parte lavorato in passato ed il cui principale compito é quello di comunicare in user space l'esito della trasmissione di ogni singolo frame 802.11 verso il primo next hop, ossia l'AP. Integrando le informazioni in possesso all'Interface Selector con questi dati (relativi all'interfaccia di rete WiFi in modalitá managed) é possibile individuare con una maggior tempestivitá quando avviene una degradazione del canale WiFi, tutto questo senza dover inviare alcun pacchetto di controllo aggiuntivo. Nonostante questi nuovi dati siano relativi all'esito della trasmissione fino al solo next hop, ossia l'AP, la loro importanza non é affatto trascurabile, in quanto il punto debole in una comunicazione mista[1], in cui si concentra la probabilitá di perdita di un pacchetto, é proprio il canale wireless.
É possibile tentare inoltre un approccio cross-layer simile anche per le altre tecnologie wireless, ma a differenza del protocollo 802.11 in questo caso non é stato fatto ancora alcun studio.

8.2 Gestione TCP

Per la gestione delle connessioni TCP, sono necessarie alcune migliorie:

- Occorre rendere le finestre di trasmissione (le corrispondenti sliding windows del protocollo TCP) di dimensione variabile, in modo da adattarle alla qualitá del canale, cosí da evitare l'invio di numerosi messaggi quando il canale in uso presenta una qualitá non sufficiente a sostenere in maniera efficiente il traffico in questione.

[1] Con il termine mista si intende una comunicazione in parte wireless (solitamente tra MN e AP) e in parte su cavo (solitamente tra AP e il resto di Internet).

- Occorre implementare un meccanismo simile al controllo di congestione del TCP, in modo da evitare problemi di congestione nei vari nodi della rete che si trovano tra il MN e il proxy server.

8.3 Scenari d'uso alternativi

Gestore della rete partecipe alla scelta dell'interfaccia Nella descrizione delle varie politiche di scelta dell'interfaccia da usare il roulo centrale é sempre stato del MN. É comunque possibile, previa qualche modifica, permettere al proxy server di partecipare alla scelta dell'interfaccia. I motivi per cui un tale scenario potrebbe essere plausibile sono due:

- Il proxy server potrebbe essere a conoscenza di informazioni quali la copertura fornita dai vari provider 3G e WiMax. Queste informazioni, assieme alla posizione del MN, potrebbero essere utilizzate per la valutazione della qualitá del canale, riducendo sensibilmente l'invio di pacchetti di controllo sulle interfacce 3G e WiMax quando a priori si sa che nel punto in cui si trova il MN la copertura é scarsa. In questo modo, oltre a ridurre l'overhead e diminuire la dispersione di energia dovuta alla trasmissione, lo stesso utente potrebbe beneficiarne con un risparmio in termini economici. Infatti 3G e WiMax tipicamente hanno tariffe che si basano sulla quantitá di traffico o sulla durata della connessione.

- Nel caso in cui uno stesso provider di rete fornisca servizi per tutte le tecnologie, potrebbe risultare utile dal punto di vista del provider che il proxy server costringa o consigli ad un determinato MN di utilizzare se possibile un'interfaccia invece di un'altra, perché il provider deve diminuire il carico di lavoro su una delle infrastrutture.

HPforMSC solo per determinate applicazioni HPforMSC é nato come architettura per la gestione della mobilitá general purpose, nel senso che é stato progettato per offrire servizi a tutte le comunicazioni del MN verso

l'esterno, indipendentemente dall'applicazione e dalla tipologia di traffico. Ció non toglie che per motivi di marketing o comunque economici non sia uno stesso provider di servizi quale per esempio Skype, Hulu, Netflix[2], iCloud, Dropbox ... a voler fornire un sostegno alla mobilitá ai propri clienti sostenendo l'onere della gestione di proxy server dedicati al solo traffico diretto verso il provider stesso. In questo caso sul MN il Proxy Client lascerebbe inalterato tutto il traffico estraneo al provider in questione, mentre redirigerebbe il solo traffico riguardante il provider verso il proxy server, il quale a sua volta, appartenendo al provider del servizio, potrebbe anche contenere al suo interno i dati richiesti dall'utente, in modo da diminuire la latenza end-to-end ed annullare i ritardi dovuti ad una deviazione nella route dei pacchetti.

[2]Netflix e Hulu offrono servizi di video on demand negli Stati Uniti.

Conclusioni

Negli ultimi anni la diffusione di device mobili e la costante ricerca di connettivitá da parte delle persone ha spinto il mondo della ricerca ad individuare nuove soluzioni per poter rendere possibili connessioni con buone prestazioni offrendo al tempo stesso un supporto alla mobilitá per garantire alle persone una connessione ad Internet ovunque ed in ogni momento.
HPforMSC é appunto un'architettura per il supporto alla mobilitá che sfrutta le potenzialitá di tutte le interfacce di rete a disposizione di un device mobile, in modo da mitigare le limitazioni di ciascuna tecnologia sfruttando i punti di forza delle altre.
HPforMSC, grazie ad un'architettura basata su hidden proxy, permette in maniera del tutto trasparente all'utente di utilizzare contemporaneamente le varie interfacce di rete a disposizione mantenendo inalterate le performance di tutte le comunicazioni in essere durante un handover, sia esso orizzontale che verticale.

Bibliografia

[1] C. Benvenuti, *Understanding Linux Network Internals*, O'Reilly, 2005.

[2] Devarapalli, et al, *Network Mobility (NEMO) Basic Support Protocol*. RFC 3963, January 2005.

[3] D. Johnson, C. Perkins, J. Arkko, *Mobility support in IPv6*. RFC 3775 June 2004.

[4] R. Koodli, *Fast Handover for Mobile IPv6*. IETF RFC 4068, July 2005.

[5] H. Soliman et al., *Hierachical Mobile IPv6 Mobility Management (HMIPv6)*. IETF RFC 4140, Aug. 2005.

[6] S. Gundavalli et al., *Proxy Mobile IPv6*. IETF Internet Draft, draft-ietf-netlmm-proxymip6-01.txt, June 2007.

[7] K. Kong et al, *Mobility management for All-IP mobile networks: Mobile IPv6 vs. Proxy Mobile IPv6*. IEEE Wireless Communications, April 2008.

[8] Wakikawa (Ed.), et al, *Multiple Care-of Addresses Registration*. IETF Internet-Draft, May 2009.

[9] E. Perera, V. Sivaraman, A. Seneviratne, *Survey on network mobility support*. Mobile Computing and Communications Review 8 (2), 2005.

[10] D. Meyer, *The Locator/Identifier Separation Protocol (LISP)*. February 2008, disponibile su http://www.lisp4.net/documentation/extensive-lisp-overview/ .

[11] D. Meyer, D. Lewis, D. Farinacci, *LISP Mobile Node* IETF Internet-Draft, October 2010, disponibile su http://tools.ietf.org/html/draft-meyer-lisp-mn-04#page-11 .

[12] D. Farinacci, V. Fuller, D. Meyer, D. Lewis, *Locator/ID Separation Protocol (LISP)*, IETF Internet-Draft. March 2011, disponibile su http://tools.ietf.org/html/draft-ietf-lisp-11.

[13] R. Moskowitz, P. Nikander, *Host Identity Protocol (HIP) Architecture*, IETF RFC 4423, May 2006.

[14] F. Teraoka, *LIN6: A Solution to Multihoming and Mobility in IPv6*, IETF Internet Draft, 2006.

[15] E. Nordmark, M. Bagnulo, *Shim6: Level 3 Multihoming Shim Protocol for IPv6*. RFC 5533 June 2009.

[16] E. Kooler et al., *Datagram Congestion Control Protocol (DCCP)*, IETF RFC 4340, March 2006.

[17] M. Riegel, M. Tuexen, *Mobile SCTP*, IETF Internet draft, Oct. 2006.

[18] J. Rosemberg et al., *SIP: session initiation protocol*, IETF RFC 3261, June 2002.

[19] H. Schulzrinne, E. Wedlund. *Application-layer mobility using SIP* SIGMOBILE Mob. Comput. Commun. Rev. 4, 3 July 2000, 47-57.

[20] T.M. Lim, Chai Kiat Yeo, Francis Bu Sung Lee, Quang Vinh Le, *TM-SP: Terminal Mobility Support Protocol*, IEEE Transactions on Mobile Computing, vol. 8, no. 6, pp. 849-863, June 2009.

[21] Udugama, A.; Kuladinithi, K.; Gorg, C.; Pittmann, F.; Tionardi, L.; *NetCAPE: Enabling Seamless IMS Service Delivery across Heterogeneous Mobile Networks*, Communications Magazine, IEEE , vol.45, no.7, pp.84-91, July 2007.

[22] J. Rosenberg, J. Weinberger, C. Huitema, R. Mahy, *STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)*, RFC 3489, March 2003.

[23] J. Rosenberg, R. Mahy, C. Huitema, *Traversal Using Relay NAT (TURN)*, Internet-Draft, September 2005.

[24] S. Salsano et al., *SIP-based mobility management in next generation networks*, IEEE Wireless Communications, pp. 92-99, April 2008.

[25] V. Ghini, G. Lodi, F. Panzieri, *Always Best Packet Switching:the Mobile VoIP Case Study*, Journal of Communications, vol. 4, no. 9, October 2009.

[26] V. Ghini, S. Cacciaguerra, F. Panzieri, P. Salomoni, *An hidden proxy for seamless & ABC multimedia mobile blogging.* Proc. of IEEE Consumer Communications & Networking, 2nd IEEE International Workshop on Networking Issues in Multimedia Entertainment (NIME2006), Las Vegas, NV (USA), January 2006.

[27] H. Fathi, R. Prasad, and S. Chakraborty, *Mobility Management for VoIP in 3G Systems: Evaluation of Low-Latency Handoff Schemes*, IEEE Wireless Comm. Magazine, vol. 12, no. 2, pp. 96-104, Apr. 2005.