Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso si Laurea Magistrale in Informatica

Confronto simulativo tra architetture per la mobilità: analisi simulatore LISP e confronto con ABPS

Relatore: Chiar.mo
Dott. Vittorio Ghini

Presentata da:
Convertino Edoardo

Sessione III

Anno Accademico 2014/2015

Appendice

Capitolo 1

Introduzione

1.1 Introduzione

L'utilizzo sempre crescente di dispositivi mobili, lo sviluppo di applicazioni *mobile* in continuo aumento, e la necessità di una sempre migliore qualità della comunicazione, ha portato grande interesse ad analizzare i protocolli di supporto alla mobilità dei terminali. Questi, tra i quali il più conosciuto è forse Mobile IP, vengono posti in esame utilizzando diverse metriche per valutarne le prestazioni. Si confrontano dunque due di questi protocolli: LISP e ABPS; per ognuno dei quali ne viene presentata e descritta l'architettura e le principali funzionalità; entrambe queste architetture per il supporto alla mobilità, prevedono delle specifiche per fornire continuità nella comunicazione durante il roaming di un nodo multihomed. Vengono presentati poi gli strumenti con i quali verrà effettuata il l'analisi: il simulatore a eventi discreti OMNeT++ e il suo framework INET. Successivamente sono descritte le principali componenti dei simulatori per LISP e ABPS, che modellano le meccaniche dei due protocolli analizzati. Questi sono stati sottoposti a modifiche mirate a correggerne eventuali anomalie di comportamento, e ad introdurre nuove funzionalità, soprattutto per quanto riguarda ABPS, che era solo parzialmente implementato. Sono mostrati gli scenari in cui verranno effettuati i test per il confronto delle prestazioni: uno scenario semplice e uno che cerca di proporre una rete urbana verosimile; di seguito vengono elencati i parametri e le configurazioni utilizzate per ognuno dei due scenari. Infine vengono presentati i risultati mettendo a confronto due aspetti della mobilità dei terminali: durata dell'intervallo di indisponibilità e latenza dei pacchetti.

Capitolo 2

LISP

2.1 Introduzione

Il concetto di LISP (Locator ID Separation Protocol) è stato inizialmente una proposta in tema di adattamento e di scalabilità, dei sistemi di routing e indirizzamento di internet. Questi infatti, non scalano in maniera soddisfacente di fronte alla crescita vertiginosa del numero di siti presenti nella rete. Una delle principali ragioni è il grande numero di questi con più connessioni ad internet (multihomed), e che non possono essere indirizzati in relazione ad un prefisso di aggregazione basato sulla topologia o sul provider. Utilizzare un singolo indirizzo per l'identificazione e la collocazione topologica richiede ottimizzazione su due assi contrastanti: perchè l'instradamento sia efficiente l'indirizzo deve essere assegnato topologicamente, mentre perchè sia semplice gestire i device dopo un cambiamento topologico senza doverli rinumerare, l'indirizzo non dovrebbe essere legato in alcun modo alla topologia della rete.

L'approccio utilizzato da LISP per risolvere questo problema è di sostituire l'indirizzo IP con due nuovi tipi di numerazioni: Routing Locators (RLOCs) e Endpoint Identifiers (EIDs), i primi sono assegnati ai nodi di connessione alla rete (attachment points) ed utilizzati per l'instradamento e il forwarding dei pacchetti attraverso la rete, i secondi sono assegnati indipendentemente dalla topologia, sono utilizzati per identificare i device e sono suddivisi seguendo dei vincoli amministrativi. RLOCs e EIDs sono sintatticamente identici agli indirizzi IP comunemente utilizzati per l'instradamento; è la semantica con la quale vengono utilizzati che li differenzia.

2.2 Architettura e funzionamento

Il concetto chiave sul quale si basa l'architettura di LISP è il seguente: gli end-system devono continuare a operare esattamente nello stesso modo in cui operano adesso. Gli indirizzi IP che gli host utilizzati non devono cambiare. Nella terminologia LISP, questi indirizzi (quelli degli endpoint) rappresentano degli EIDs. La maggior parte dei router lungo il percorso tra due host non cambierà; essi continueranno a instradare e inoltrare pacchetti in base all'indirizzo di destinazione. Gli host inviano solo ad indirizzi EID. Essi non sanno in che modo avviene l'indirizzamento (EID-RLOC mapping) ma assumono che i pacchetti arriveranno alla loro destinazione finale.

Una componente fondamentale e cardine dell'intera architettura LISP è il "Tunnel Router". Un Tunnel Router ha la funzione principale di aggiungere un header LISP ad un pacchetto originato da un host del sito e/o di rimuoverlo prima della consegna all'host di destinazione. Esistono però in LISP due tipi di Tunnel Router:

- Ingress Tunnel Router (ITR)
  Un ITR è un router che si trova in un sito LISP. I pacchetti inviati da host all'interno del sito a host all'esterno del sito possono essere incapsulati dall'ITR. Questo gestisce l'IP di destinazione come se fosse un EID ed effettua una ricerca per il mapping EID-RLOC. Il router quindi aggiunge un header IP con uno degli RLOC a lui assegnati come indirizzo sorgente e il risultato del mapping come indirizzo di destinazione. Questo RLOC può essere anche un intermediario, un proxy più vicino all EID di destinazione, che saprà effettuare il mapping. Genericamente, un ITR riceve i pacchetti IP dagli host del sito al quale appartiene da un lato, e invia pacchetti IP incapsulati LISP verso internet dall'altro.

- Egress Tunnel Router (ETR)

  Un ETR è un router che accetta pacchetti IP, dove l'indirizzo di destinazione nell'header più esterno è uno degli RLOC ad esso assegnati. Questo rimuove l'header, e inoltra il pacchetto verso l'indirizzo IP di destinazione trovato nell'header interno. Genericamente un ETR riceve pacchetti IP incapsulati LISP da un lato, e invia pacchetti IP decapsulati agli host del sito dall'altro. Anche un server host può essere un endpoint per un tunnel LISP.

Quindi per i router tra l'host sorgente e l'ITR, così come per i router tra l'ETR e l'host di destinazione, gli indirizzi di destinazione dei pacchetti IP saranno EID. Per i router tra ITR e ETR invece, gli indirizzi di destinazione IP saranno RLOC.

In assenza di un sistema di mapping da EID a RLOC che garantisca la possibilità di una comunicazione inter-site, per i pacchetti provenienti da un host non è prevista la consegna end-to-end, possono essere utilizzati solo per comunicazioni locali intra-site. In Figura 1 viene mostrato un esempio di come può essere strutturata l'architettura LISP; l'immagine mostra bene la separazione tra spazio EID e spazio RLOC; il primo dominio non è routabile dalla rete di interconnessione ed appartenente ai device, mentre il secondo è indstradabile e appartenente ai router. Al centro della rete sono collocati Map Resolver e Map Server, le interfacce che l'architettura e le componenti LISP utilizzano per usufruire del Mapping System distribuito; questo sistema di mapping può essere uno tra i tanti proposti per il supporto di LISP.

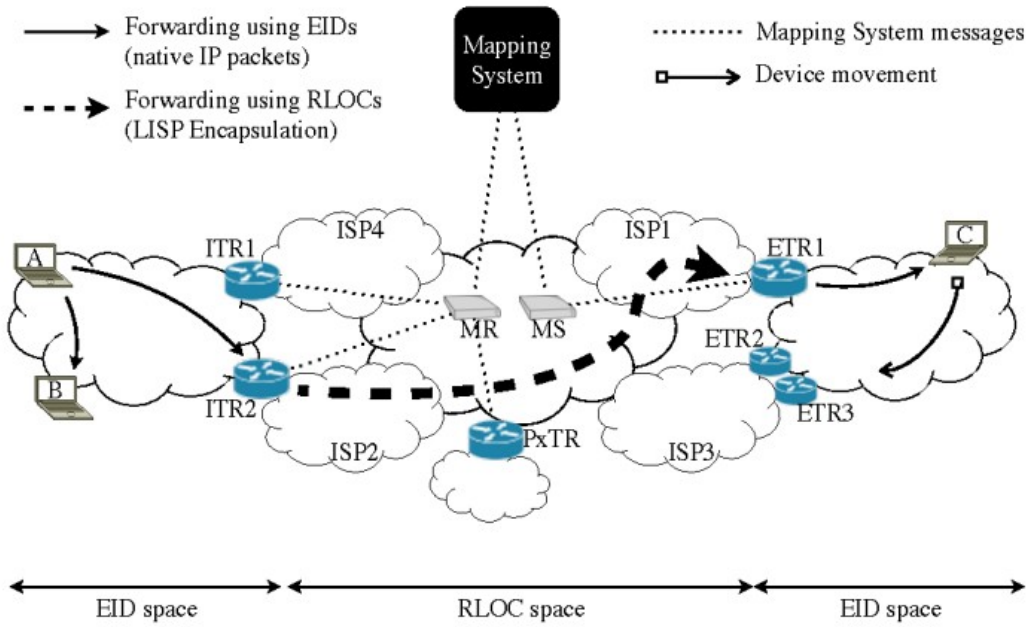


Figura 1 - Implementazione sul nodo mobile.

2.2.1 Componenti xTR per il mapping EID-to-RLOC

ITR ed ETR (spesso queste due funzioni possono essere ricoperte dallo stesso dispositivo) hanno una componente ciascuno che gli permette di poter partecipare alla risoluzione del mapping EID-to-RLOC. Queste componenti sono le seguenti:

- EID-to-RLOC Cache

  è una cache di breve durata in un ITR, che memorizza al suo interno i mapping EID-to-RLOC precedentemente richiesti al database di mapping. Questa cache è distinta dal database per il mapping EID-to-RLOC; è dinamica e relativamente piccola, mentre il database di mapping è distribuito, relativamente statico e globale.

- EID-to-RLOC Database

è un database globale e distribuito che contiene tutti i mapping dei prefissi EID-to-RLOC conosciuti. Ogni ETR contiene una piccola parte di questo database: i mapping per i prefissi EID che si trovano in quel sito. Le stesse mapping entry devono essere configurate su tutti gli ETR di un determinato sito. In uno stato stabile del database, i prefissi EID per il sito e i locator-set per ogni prefisso EID devono essere gli stessi su tutti gli ETR all'interno del sito.

2.2.2 Aggiornamento EID-to-RLOC Mapping

L'architettura LISP utilizza il caching per ottenere e memorizzare i mapping EID-to-RLOC, quindi l'unica via per aggiornare questi mapping è quella di richiedere la mapping entry non più valida. In ogni caso, l'ITR non sa quando i mapping cambiano, e gli ETR non tengono traccia di quali ITR hanno richiesto i loro mapping. Per ragioni legate alla scalabilità, si vuole mantenere questo approccio; per mantenerlo però è necessario fornire una via in cui gli ETR possano cambiare i loro mapping, e informare i siti che attualmente stanno utilizzando questi per comunicare. Per l'aggiornamento delle cache dei vari ITR esistono tre meccanismi, uno operazionale (Clock Sweep) e due protocollari (SMR e Map-Versioning).

2.2.2.1 Clock Sweep

Questo approccio sfrutta la pianificazione in anticipo ed utilizza un count-down per le TTL, per definire quando un mapping in cache deve essere aggiornato. Di default il TTL è impostato a 24 ore, quindi c'è una finestra di 24 ore per fare scadere i vecchi mapping. Di seguito è riportato un esempio di Clock Sweep:

- 24 ore prima che il cambiamento di mapping abbia effetto, un amministratore della rete configura gli ETR del sito per iniziare la finestra di clock sweep.

- durante questa finestra di tempo, gli ETR continuano a inviare Map-Reply con i mapping attuali ancora non aggiornati, impostando però il TTL ad 1 ora per questi ultimi.

- 24 ore dopo, tutte le precedenti entry della cache sono scadute, ed ogni entry attiva scadrà in massimo un'ora. Durante quest'ora di tempo, gli ETR continuano a inviare Map-Reply con i mapping attuali ancora non aggiornati, impostando però il TTL ad 1 minuto per questi ultimi.

- allo scadere della finestra di un'ora, gli ETR invieranno le Map-Reply con i mapping aggiornati. Così, ogni cache può ottenere i nuovi mapping immediatamente, o nel giro di un minuto se ha ancora i mapping in cache. I nuovi mapping sono inseriti in cache con un TTL equivalente a quello della Map-Reply che glieli ha forniti (default 24 ore).

2.2.2.2 Solicit-Map-Request

L'ETR non tiene traccia degli ITR che utilizzano i mapping da lui forniti, e quindi non può sapere quali ITR hanno bisono di ricevere i mapping aggiornati una volta che questi cambiano, sull'ETR stesso. Quindi questo invierà una Solicit-Map-Request (SMR) a quei siti da cui ha ricevuto pacchetti nell'ultimo minuto. In particolare, un ETR invierà un SMR a un ITR che ha recentemente inviato dati incapsulati LISP. Un SMR non è altro che una Map-Request con un bit settato come flag. Un ITR o PITR invierà una Map-Request quando riceve una SMR. Sia il mittente del SMR che quello della successiva Map-Request devono impostare un limite al rate di questi messaggi. La seguente procedura mostra come avviene uno scambio SMR quando un sito sta effettuando "Locator-Set compaction" per un mapping:

- quando un mapping in un ETR cambia, gli ETR in quel sito iniziano a mandare Map-Request con SMR impostato per ogni locator in ogni entry della cache dell'ETR (gli ITR che hanno inviato fino al minuto precedente).

- l'ITR che riceve il messaggio SMR invierà una Map-Request al RLOC sorgente della SMR, o direttamente al database globale per il prefisso EID utilizzato nel messaggio SMR.

- l'ETR al sito con i mapping aggiornati risponderà alla Map-Request con la Map-Reply per quella entry con il relativo nonce. È impotante limitare le Map-Reply per evitare di congestionare la rete.

- gli ETR al sito con i mapping aggiornati, registrano che il sito che ha inviato la Map-Request ha ricevuto i nuovi mapping nella Map-Cache con i relativi Locator-Status-Bits. L'ETR dunque interrompe l'invio dei messaggi SMR.

2.2.2.3 Map-Versioning

Un ETR, quando invia le Map-Reply, inoltra anche il proprio numero di Map-Version. Questo è conosciuto come Destination Map-Version Number. L'ITR include questo numero nei pacchetti che incapsula verso il sito dell'ETR. L'ETR che decapsulerà il pacchetto e rileva che il Destination Map-Version Number inviato dal ITR è inferiore alla versione corrente del mapping, darà avvio alla procedura di Solicit-Map-Request come descritto in 2.2.2.2.

Un ITR, quando incapsula i pacchetti per l'ETR, può inviare il proprio Map-Version Number, conosciuto come Source Map-Version Number. Quando un ETR decapsulando il pacchetto rileva che la Source Map-Version è maggiore,

dell'ultimo inviato in una Map-Reply da sito di quel ITR, l'ETR invierà una Map-Request ad uno degli ETR di tale sito per aggiornare la propria Map-Version.

2.2.3 Packet Flow Sequence

Il seguente è un esempio di come un client (host1.abc.example.com) che comunica con un server (host2.xyz.example.com) interagisca ed utilizzi l'architettura LISP per inviare il pacchetto a destinazione:

- host1 vuole aprire una connessione TCP con host2; esegue un lookup DNS per host2.xyz.example.com, gli viene restituito l'EID di destinazione relativo a host2. L'indirizzo locale di host1 è utilizzato come indirizzo EID sorgente. Il pacchetto è inviato attraverso il sito fino a che non raggiunge l'ITR LISP.

- l'ITR deve essere in grado di mappare l'EID di destinazione specificato nel pacchetto su un RLOC di uno degli ETR al sito di destinazione, e quindi, se non è già presente la entry nella cache locale, l'ITR invia una Map-Request richiedendo un mapping EID-to-RLOC per l'EID destinazione al database utilizzato (3.3 LISP+ALT).

- quando non viene utilizzato un sistema di mapping alternativo, la Map-Request viene instradata attraverso il sistema di routing sottostante. Altrimenti la Map-Request è inviata al sistema database di mapping, LISP+ALT ad esempio. In entrambi i casi, quando la Map-Request raggiunge l'ETR al sito di destinazione, questo processerà il pacchetto come messaggio di controllo.

- l'ETR preleva l'EID di destinazione della Map-Request e lo riscontra tra i prefissi presenti nel suo EID-to-RLOC mapping database locale. Questo è una lista dei prefissi EID che l'ETR è in grado in instradare e che

risiedono all'interno del sito LISP. Se non vi sono riscontri, la Map-Request è scartata. Altrimenti, è inviata una Map-Reply di ritorno all'ITR.

- l'ITR, ricevuta la Map-Reply, estrae le informazioni di mapping ottenute dal pacchetto. Queste informazioni sono memorizzate nella EID-to-RLOC mapping cache locale dell'ITR.

- di conseguenza i pacchetti da host1 a host2 avranno un header LISP aggiunto dall'ITR, usando l'RLOC appropriato come destinazione appreso precedentemente dall'ETR. Il pacchetto può essere inviato ad un ETR diverso da quello che ha inviato la Map-Reply, a causa di politiche di hashing del sito sorgente o di politiche riguardo i locator-set del sito di destinazione.

- l'ETR riceve i pacchetti direttamente (siccome il pacchetto è incapsulato utilizzando uno dei suoi RLOC come indirizzo di destinazione), rimuove l'header LISP e inoltra il pacchetto al destinatario all'interno del sito LISP.

2.3 Mapping System

LISP assume che vi sia un database che registri e propaghi i mapping EID-RLOC globalmente. L'integrazione del sistema di mapping prevede l'introduzione di due componenti in LISP, che hanno funzionalità simili a quelle di un sistema DNS e che forniscono un "font-end" semplificato per interfacciarsi al database di mapping EID-to-RLOC:

- Map Server

  Una componente dell'infrastruttura che attraverso il meccanismo di registrazione per i prefissi EID (o altre fonti autoritative se esistono), viene a conoscenza delle mapping entries per questi da un ETR.

Successivamente il Map Server pubblica quei prefissi EID in un database di mapping. Una volta che il Map-Server ha i prefissi EID registrati per i vari client ETR, può accettare e processare Map-Request per questi (inoltrate attraverso il sistema di mapping da un Map-Resolver).

- Map Resolver

  Una componente dell'infrastruttura che accetta Map-Request incapsulate LISP da un ITR (tipicamente), e determina se l'indirizzo IP di destinazione fa parte del namespace degli EID; nel caso non ne facesse parte, risponde con una Negative Map-Reply. Altrimenti, il Map-Resolver trova il mapping EID-to-RLOC appropriato consultando il sistema di database di mapping sul quale fa affidamento tramite messaggi Map-Request.

2.3.1 Registrazione al Map-Server

Un ETR pubblica i propri prefissi EID in un Map-Server inviando messaggi Map-Register LISP. Questi messaggi sono inviati periodicamente da un ETR al Map-Server con un intervallo suggerito di un minuto. Un Map-Server dovrebbe rimuovere la entry per un ETR se non ha ricevuto una Map-Register valida nei tre minuti precedenti. Un ETR può richiedere esplicitamente un acknowledge di ricezione e processamento di una Map-Register impostando il flag "want-map-notify"; un Map-Server che riceve una Map-Register così configurata, risponderà con un messaggio di Map-Notify.

2.3.2 Richiesta di EID-to-RLOC

Un ITR è configurato con uno o più indirizzi di Map-Resolver a cui può fare riferimento. Questi indirizzi sono Locator (RLOCs) e devono essere indirizzabili dalla rete sottostante; questi non deono aver bisogno di essere risolti attraverso il mapping EID-to-RLOC LISP, in quanto verrebbe introdotta

dipendenza circolare. Quando utilizza un Map-Resolver, l'ITR non ha bisogno di connettersi ad alcun altro sistema di database di mapping, in quanto è tutto gestito dall'interfaccia Map-Resolver. In particolare, l'ITR non deve connettersi all'infrastruttura LISP+ALT o implementare i protocolli BGP (Border Gateway Protocol) e GRE (Generic Routing Encapsulation) che essa usa.

Un ITR invia una Map-Request incapsulata al Map-Resolver quando necessita di un mapping EID-to-RLOC che non trova nella propria map-cache locale. L'utilizzo del Map-Resolver riduce notevolmente sia i costi di complessità di implementazione dell'ITR e i costi associati alla sua attività. In risposta ad una Map-Request incapsulata, l'ITR può ricevere uno dei seguenti messaggi:

- Negative Map-Reply con action code "Natively-Forward" (TTL di 15 minuti) dal Map-Resolver se questo può determinare che l'EID richiesto non esiste. L'ITR salva il prefisso EID allegato nella Map-Reply nella cache, segnandolo come indirizzo non-LISP, sapendo di non dover incapsulare i pacchetti con quella destinazione.

- Negative Map-Reply con action code "Natively-Forward", dal Map-Resolver che è autoritativo per quel prefisso EID, il quale riconosce l'EID richiesto ma non ha registrato nessuna entry. In questo caso, l'EID richiesto è detto un "buco" nel prefisso EID autoritativo.

- LISP Map-Reply dal ETR che possiede il mapping EID-to-RLOC, o possibilmente dal Map-Server che risponde in vece del ETR. Se il Map-Resolver non ha abbastanza informazioni ma conosce che l'EID esiste, deve inoltrare la Map-Request ad un altro device che ha pià informazioni riguardo all'EID richiesto. Per fare questo, inoltra la Map-Request incapsulata, con l'indirizzo RLOC del ITR originale come sorgente, al sistema di database di mapping (ad esempio LISP+ALT).

2.3.3 LISP+ALT

LISP+ALT (Alternative Logical Topology) è un'architettura push/pull ibrida. Viene effettuato l'advertise dei prefissi EID aggregati tra i router-ALT e i rari ITR che sono direttamente connessi ad ALT tramite tunnel e BGP. Mapping EID-to-RLOC specifici sono richiesti da un ITR (e restituiti da un ETR) usando LISP quando questo invia una richiesta tramite Map-Resolver, o tramite un router ALT di bordo.

L'idea di base in LISP+ALT è di creare una rete virtuale soprastante l'attuale rete di internet utilizzando il tunneling (GRE) per interconnettere i router ALT. I router ALT si interconnettono utilizzando BGP, propagando gli aggiornamenti dei prefissi EID tra di essi. Le informazioni riguardanti i prefissi EID sono acquisite dagli ETR al bordo di ALT o con l'utilizzo dell'interfaccia Map-Server (nel caso comune), o tramite configurazione statica, o da ETR collegati tramite BGP.

I Map-Resolver ottengono i percorsi verso i Map-Server attraverso ALT, per i prefissi EID richiesti. Un ITR normalmente usa un Map-Resolver per inviare i propri datagram ALT nella rete ALT, ma può utilizzare anche un percorso ALT statico di default o connettersi ad ALT utilizzando BGP. In maniera analoga, un ETR che normalmente registra i propri prefissi al database di mapping usando il Map-Server può a volte connettersi direttamente ad ALT utilizzando BGP.

Per facilitare l'aggregazione dei prefissi EID e dunque la scalabilità del sistema di mapping ALT, la topologia BGP di ALT è disposta gerarchicamente; l'assunzione che ogni collegamento tra nodi interni alla rete sia un tunnel, permette alla topologia di seguire la gerarchia di assegnamento dei prefissi EID. La rete è provvista di collegamenti ridondanti per compensare il crash di

un nodo o di un collegamento: finchè i router sono in funzionamento, la rete sottostante provvederà a fornire instradamenti alternativi per mantenere i tunnel attivi e la connettività BGP tra i router ALT. In Figura 2 sottostante, è mostrato com'è relazionata la rete di interconnessione ALT con quelli che sono i siti LISP, e come questa inoltra le richieste ai Map Server degli ETR di destinazione.

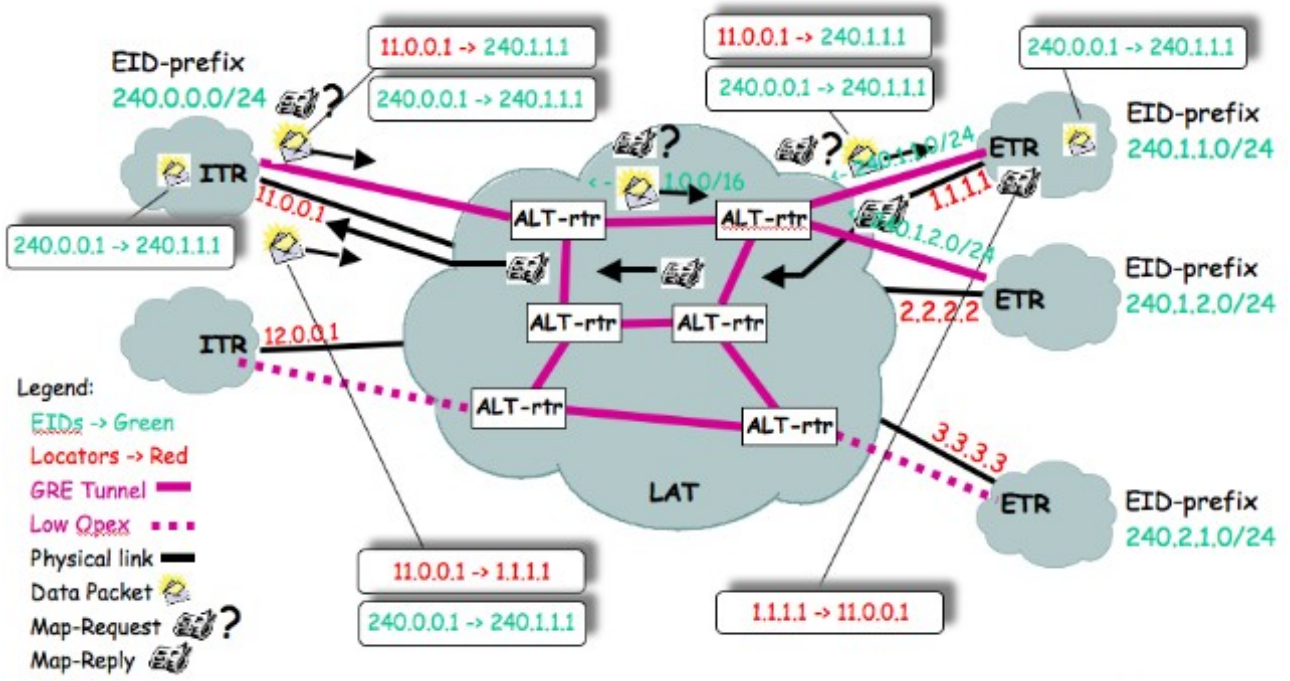


Figura 2 - Interazione tra LISP e la rete di router ALT.

2.4 Comunicazione con siti non-LISP

Ci sono due casi di instradamento unicast in uno scenario in cui è impiegata l'architettura LISP nei quali è necessario/possibile introdurre due nuove componenti di rete:

- sito LISP invia a sito non-LISP

  in questo caso il sito LISP può inviare i pacchetti direttamente al sito

non-LISP, in quanto i suoi prefissi sono instradabili, ed il sito non-LISP non deve implementare nulla di nuovo per poter ricevere questi pacchetti. L'unica cosa che un sito LISP deve tenere in considerazione è quando incapsulare LISP e quando invece instradare direttamente il pacchetto: l'ITR conosce esplicitamente che la destinazione è non-LISP se l'IP di destinazione di un pacchetto riscontra una Map-Cache entry negativa, che ha come azione "Natively-Forward".
Si possono verificare alcune situazioni dove i pacchetti originati da un sito LISP potrebbero non essere inoltrati direttamente verso un sito non-LISP. In questi casi si fa affidamento sull'utilizzo di un Proxy Egress Tunnel Router (PETR).

- sito non-LISP invia a sito LISP

  quando un nodo in un sito non-LISP invia un pacchetto dove il destinatario è un indirizzo EID, questo è inoltrato all'interno del sito di origine finchè non raggiunge un router che non è in grado di instradarlo ulteriormente, il quale successivamente scarterà il pacchetto. Per evitare di scartare questi tipi di pacchetti viene introdotto il Proxy Ingress Tunnel Router (PITR).

- PETR

  il PETR permette a siti LISP di inviare pacchetti a siti non-LISP in caso in cui, l'access network di alcuni provider non permetta al sito LISP di inviare i pacchetti con l'EID come indirizzo IP sorgente. La presenza di questa componente evita al ITR di dover decidere se incapsulare o non incapsulare LISP i pacchetti, consentendogli di incapsularli sempre. I pacchetti destinati a siti non-LISP verranno incapsulati ed instradati verso il PETR del sito di origine del pacchetto.

- PITR

  il PITR permette a siti non-LISP di inviare pacchetti a siti LISP altrimenti non instradabili. È una componente della rete che condivide molte caratteristiche con l'ITR LISP, che prevede però due funzionalità differenti, effettuare il route advertisement per gli spazi dei prefissi EID (fortemente) aggregati in vece dei siti LISP rendendoli raggiungibili dai siti non-LISP, e incapsulare il traffico non-LISP in pacchetti LISP che instrada verso l'RLOC di destinazione.
  Per suddividere il carico di pacchetti instradati dai vari PITR verso i siti LISP e minimizzare lo scope del route advertisement, questi dovrebbero essere posizionati nella rete in prossimità dei vari siti non-LISP.

Proxy-ITR e Proxy-ETR possono e dovrebbero essere divisi, in quanto la collocazione del PITR è suggerita vicino al sito non-LISP per il quale fa le veci, mentre il PETR è meglio collocato vicino al sito LISP per il quale decapsula i pacchetti. Questo posizionamento asimmetrico dei due elementi della rete LISP minimizza la distanza percorsa dal flusso di pacchetti verso i due elementi, e permette di fare coarse route advertisment degli EID per i router di internet.

2.5 Lisp-MN

Nell'architettura LISP viene introdotto un elemento di rete specifico che risolvere parte delle problematiche introdotte dalla mobilità dei terminali, il nodo mobile LISP, che implementa una parte delle funzionalità standard di un Ingress Tunnel Router e di un Egress Tunnel Router; i presupposti allo sviluppo di questa componente sono i seguenti:

- permettere alle connessioni TCP di rimanere attive mentre si effettua roaming.
- permettere al nodo mobile di comunicare con altri nodi mobili mentre entrambi effettuano roaming.

- permettere al nodo mobile di utilizzare più interfacce concorrentemente (multi-homing).
- permettere al nodo mobile di essere un server, che ogni nodo mobile o nodo fisso possa trovare e connettervi.
- fornire il cammino bidirezionale più corto tra un nodo mobile ed un nodo fisso/mobile.
- non deve essere richiesto home-agent o foreign-agent.
- non deve essere richiesto l'IPv6 extension header per non incorrere in routing triangolare.

La struttura del nodo mobile LISP (LISP-MN) è simile a quella di un sito LISP con funzionalità di ITR ed ETR. Il nodo mobile però incapsula LISP solamente i pacchetti non locali e destinati a siti non-LISP al Proxy-ETR. Quando il nodo mobile LISP entra in una nuova rete, riceve un nuovo RLOC, in quanto è l'ETR autoritativo per il suo prefisso EID. Questo deve registrare il suo set di RLOC aggiornato (Map Register). Le nuove sessioni possono essere inizializzate non appena il nodo avrà teminato la procedura di registrazione. Quindi il nodo mobile deve aggiornare gli ITR e i PITR che possiedono un mapping non più valido per quell'EID, può fare affidamento su tutte le tecniche descritte in 2.2.2, quali SMR e Map-Versioning. È anche possibile configurare diversamente i TTL delle Map-Reply inviate dal nodo mobile, in maniera tale che possano essere rilevati i cambiamenti nel mapping.

Per supportare il multihoming e la continuazione della comunicazione durante il roaming, è suggerito da [2] di inserire (nell'implementazione di Lisp-MN) un sub-layer tra il livello IP e quello datalink; questo livello dovrebbe inoltre inviare una Map Register al Map Server ogni volta che viene configurata un'interfaccia del nodo mobile con un nuovo indirizzo IP. E partendo dalla

nozione che il nodo mobile LISP ha le stesse funzionalità di base di un ITR ed ETR, questo, una volta che cambia il suo set di RLOC, dovrebbe iniziare la procedura di aggiornamento delle cache remote con SMR come descritto in 2.2.2.2, andando ad inviare a tutti gli ITR per cui stava decapsulando pacchetti, una Solicit Map Request.

Capitolo 3

ABPS

3.1 Introduzione

La generazione attuale di dispositivi di comunicazione mobili offre diversi modi di accedere ad Internet. I terminali mobili sono dodati di più interfacce di rete (NICs) che permettono al dispositivo di connettersi tramite diversi tipi di infrastrutture di rete wireless, come 3G e Wi-Fi.

Un nodo mobile (MN) dotato di più interfacce wireless eterogenee dovrebbe essere abilitato ad usare la più appropriata di queste, basandosi sul suo contesto attuale. Il termine "più appropriata" può basarsi su differenti criteri, come il costo, la copertura, il transmission rate, QoS, sicurezza e preferenze dell'utente. In altre parole, devono essere utilizzati dei meccanismi per scegliere automaticamente quale interfaccia utilizzare quando la comunicazione inizia, quando si verifica un handover verticale, o anche in relazione alle performance della comunicazione (possono essere fornite dall'interfaccia in uso) che non devono scendere sotto una certa soglia. Tra i criteri possibili, l'utilizzo di un ABC (Always Best Connected) forza l'utilizzo di una singola rete alla volta, passando da una rete all'altra solamente quando la rete alla quale si è allacciati diventa inutilizzabile. Sebbene possa sembrare ragionevole, non tenendo in considerazione il tempo necessario per configurare un interfaccia, ABC non permette l'utilizzo di tutte le interfacce del nodo mobile.

ABPS è un architettura distribuita che si propone di colmare le lacune di un approccio ABC. Consiste principalmente in un'architettura basata su proxy

distribuita, che implementa un approccio Always Best Packet Switching. Il suo obiettivo è di permettere al nodo mobile di utilizzare simultaneamente tutte le interfacce di rete disponibili così da sommare le capacità e la portata di queste. È responsabilità del MN selezionare, per ogni datagram da trasmettere , l'interfaccia migliore da usare in quel particolare momento. ABPS raggruppa un insieme di protocolli, servizi e API per poter supportare servizi multimediali basati su SIP/RTP/RTCP mentre il MN si muove attraverso reti wireless eterogenee, anche se sono amministrate da organizzazioni differenti, e anche se sono presenti firewall e sistemi NAT. Il monitor del nodo mobile controlla e configura le interfacce disponibili in background per permettere il concorrente utilizzo di queste, e per passare da una rete all'altra, in modo trasparente all'applicazione senza incorrere in costosi handshake di livello applicazione/sessione. Dunque, questa architettura permette ad ogni applicazione di sviluppare le proprio politiche per fornire agli utenti la QoS richiesta.

3.2 Architettura e funzionamento

L'architettura ABPS-SIP/RTP è stata progettata per poter soddisfare tre principali requisiti:

- garantire la continuità della comunicazione: l'architettura incorpora un servizio di ancoraggio pubblico (ABPS proxy server), indipendente dalle access network, che lavora su un server fisso (FS, Fixed Server) con un indirizzo IP statico pubblico collocato all'esterno di qualsiasi firewall o sistema NAT. Il proxy ABPS è essenzialmente un Proxy SIP e RTP modificato, che opera come un intermediario tra il nodo mobile e il suo correspondent node, e deve garantire una comunicazione senza interruzioni. Ogni nodo mobile implementa una versione modificata dei proxy agent SIP e RTP (chiamati ABPS proxy client) che mantengono un canale di comunicazione multi-path senza interruzioni con il proxy server

ABPS, facendo fronte all'eventuale presenza di un sistema NAT o di un firewall, e permettendo l'interoperabilità con applicazioni basate su SIP e RTP che girano sul nodo mobile.

- abilitare l'utilizzo simultaneo di tutte le interfacce di rete: ABPS permette di differenziare la scelta dell'interfaccia per ogni datagram, introducendo la possibilità di cambiare a seconda delle esigenze. Il meccanismo di QoS permette di ritrasmettere i datagram persi da un'interfaccia differente a quella dalla quale sono stati trasmessi la volta precedente, senza incorrere in duplicazione dei messaggi a livello applicazione. Il proxy client e server ABPS collaborano e adottano politiche di load balancing e recovery, per massimizzare il throughput, riducendo il loss rate e il costo economico. Il modello ABPS migliora il tradizionale modello ABC che richiede che il nodo mobile identifichi la migliore rete wireless tra quelle disponibili tramite le sue interfacce, e una volta individuata, questa viene utilizzata come unico punto di accesso ad internet.

- minimizzare la latenza attraverso signaling shortcuts: l'obiettivo è di minimizzare la latenza ad ogni occorrenza di un cambio nei parametri della comunicazione. Il client e il server ABPS utilizzano estensioni di SIP e RTP che implementano shortcuts dei messaggi di signaling. In particolare, queste estenzioni introducono dei campi addizionali (nei pacchetti SIP e RTP) necessari ad identificare univocamente il mittendte del messaggio (Proxy identifier) anche quando cambia indirizzo IP. SRTP (l'estensione standard) non include il Proxy identifier in quanto il flusso RTP utilizza la propria porta UDP dedicata per ogni lato della comunicazione; queste porte sono configuarte durante il setup della chiamata VoIP, e il proxy mantiene una mappa che associa le porte UDP locali e il mittente all'altro capo. Quindi, il destinatario di un messaggio

SRTP deduce l'identita del mittente usando il numero di porta sulla quale il messaggio è stato ricevuto, e verifica questa identità tramite l'Authentication Tag (parametro di sicurezza del messaggio SRTP).

Grazie a queste modifiche di SIP e RTP, il client ABPS può autonomamente decidere di inviare un datagram RTP tramite un'interfaccia differente da quella precedentemente utilizzata, senza introdurre messaggi SIP preparatori per informare il proxy server del cambio di indirizzo IP del mittente. La Figura 3 mostra un'esempio di architettura Always Best Packet Switching.

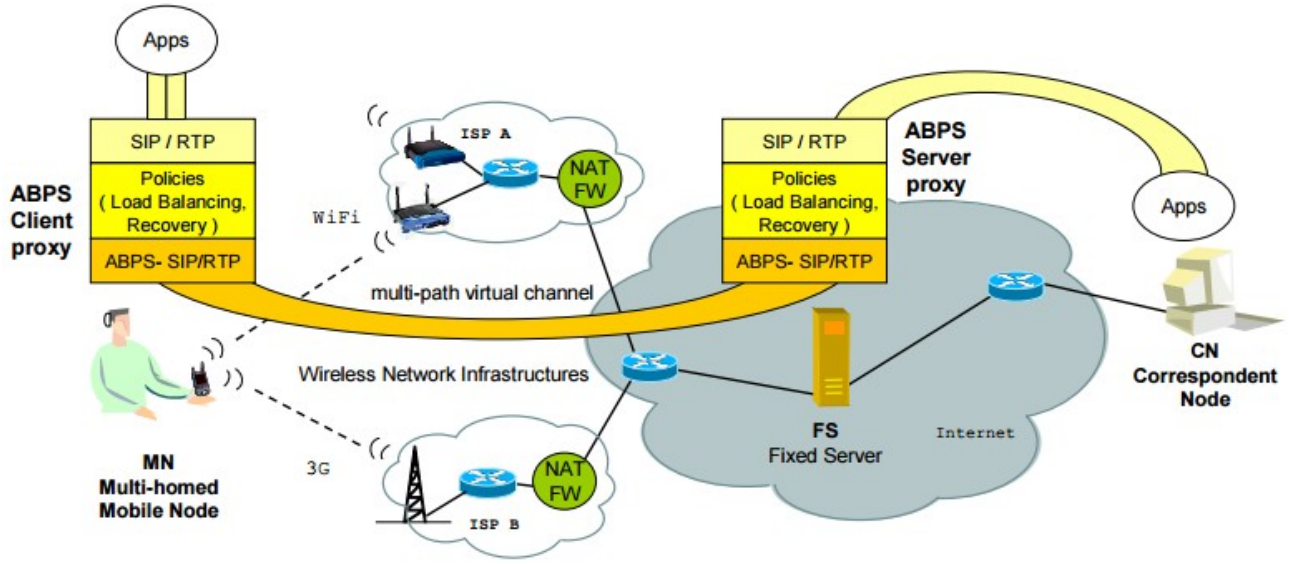


Figura 3 - Esempio di architettura ABPS.

3.2.1 Setup del sistema

Di base, l'interazione tra le differenti componenti dell'architettura esterna di ABPS si può riassumere nelle cinque fasi seguenti descritte partendo da 3.2.1.1, dove viene specificata: inizializzazione, registrazione, setup della chiamata, comunicazione e aggiornamento riguardanti l'architettura ABPS mostrata precedentemente.

3.2.1.1 Inizializzazione

La fase di inizializzazione è la *proxy client registration phase*, avviene tra il proxy client ABPS sul nodo mobile e il proxy server ABPS su un server esterno. Questa procedura registra il nodo mobile al proxy server ABPS; a questo scopo, tra i due sono scambiati il proxy client id e la chiave crittografica. Quando il nodo mobile configurerà le sue interfacce, il proxy client ABPS imposterà una sessione sicura temporanea con il proxy server ABPS utilizzando la chiave precedentemente condivisa, senza doverla ritrasmettere attraverso la rete.

Dopo questa fase il proxy client imposta un canale multi-path verso il proxy server ABPS. Un aspetto da tenere in considerazione è che il mobile node ha diversi indirizzi IP ed alcuni di questi potrebbero essere "allacciati" a reti che utilizzano firewall o sistemi NAT; queste possono cambiare gli indirizzi contenuti nei datagram IP. ABPS per "aggirare" i sistemi NAT ed i firewall che agiscono in questo modo, utilizza l'estensione SIP che definisce un nuovo parametro per il campo Via dell'header: *rport*. *rport* permette al client di richiedere al server l'invio della risposta all'indirizzo IP e alla porta dalla quale la richiesta è stata originata. Infine, il proxy client ABPS invia al proxy server ABPS il messaggio di REGISTER ABPS-SIP con il quale termina la *proxy client registration phase*.

3.2.1.2 Registrazione

Questa è la fase di registrazione in cui l'utente dell'applicazione VoIP sul nodo mobile informa il SIP server esterno che l'utente è disponibile a effettuare chiamate VoIP (*client registration phase*); il server proxy ABPS stesso può assumere il ruolo di server SIP. Tipicamente, l'utente effettua la chiamata VoIP attraverso il server SIP, usando un account SIP. Un server SIP ha il compito principale di mantenere sia lo stato (online o disconnesso), sia le informazioni (indirizzo IP attuale e porta SIP), del client SIP utilizzato dall'utente. In questo

modo un'utente può chiedere al server SIP di raggiungere un determinato utente. Un utente fornisce al server SIP le sue informazioni attraverso uno scambio di messaggi basato su SIP-REGISTER. Nell'architettura ABPS, l'utente non contatta direttamente il server SIP, ma il proxy client, questo contatta il proxy server che infine comunica son il server SIP. Le informazioni del contatto, inviate dal proxy server al server SIP, consistono in un indirizzo IP e nella porta SIP del proxy server stesso; questo permette al proxy server di prendere a tutti gli effetti il posto del nodo mobile nella comunicazione con il server SIP, nascondendo l'effettiva posizione del MN.

Infine, quando il proxy server riceve un messaggio SIP dal proxy client e ne verifica con successo l'identità del mittente tramite fingerprint, il proxy server registra l'indirizzo IP e la porta del mittente per aggiornare le informazioni riguardo la locazione del proxy client.

3.2.1.3 Setup

Questa fase può essere iniziata sia dal MN che dal CN. Per iniziare la comunicazione RTP con un utente ad un CN, un'applicazione sul nodo mobile inizia uno scambio di messaggi che coinvolge entrambi i sistemi, i due proxy ABPS e i server SIP del dominio dell'utente sul CN.

L'applicazione sul nodo mobile apre due porte UDP locali per ricevere i messaggi RTP e RTCP; poi consegna al proxy client una SIP INVITE diretta al SIP server del dominio dell'utente con il quale intende comunicare. Il proxy client ABPS consegna i messaggi al proxy server, che li inoltra al server SIP che, a sua volta, invia i messaggi all'utente destinatario. Mentre il server SIP semplicemente inoltra i messaggi, il client e il server proxy prima di ritrasmettere i messaggi ricevuti al prossimo step della comunicazione, eseguono le quattro operazioni seguenti per ottenere dall'entità seguente sia la

SIP response che i dati RTP:

- si apre una porta UDP dove si riceveranno i messaggi RTP dalla prossima entità e una per i messaggi RTCP.
- viene inserito nel messaggio SIP l'indirizzo IP come campo addizionale.
- viene sostituito, nel SDP (Session Description Protocol) incluso nel body del messaggio SIP, il numero delle porte appena aperte come numero di porte del mittente.
- vengono dunque inviati i messaggi SIP alla prossima entità attendendo la risposta SIP.

Quando il CN riceve una richiesta e l'utente accetta la chiamata, questo opera come segue: apre due porte UDP che verranno utilizzate per lo scambio di messaggi RTP e RTCP con il proxy server; imposta un messaggio SDP che includa le porte appena aperte e le opzioni selezionate per la comunicazione, include il messaggio SDP nel messaggio SIP OK e invia la risposta al server SIP aspettando per un ACK SIP che confermi che la chiamata è stabilità.
Il server SIP inoltra la risposta ricevuta dal CN direttamente al proxy server.
Quando il proxy riceve la risposta, apre due porte UDP per lo scambio di messaggi RTP e RTCP con la precedente entità nella linea di comunicazione, modifica il messaggio SDP inserendovi le porte appena aperte e invia la risposta alla entità precedente (client proxy idealmente), aspettando un ACK SIP di conferma.

3.2.1.4 Comunicazione

La fase di comunicazione è la fase dove effettivamente i dati audio vengono scambiati. Dopo una fase di configurazione, i dati RTP vengono scambiati tra il mobile node e CN attraverso tre consecutivi percorsi bidirezionali:

- il percorso tra l'applicazione MN e il proxy client ABPS, dove la

comunicazione avviene tramite protocollo RTP.

- il percorso multi-path tra il proxy client ABPS e il proxy server ABPS, dove è introdotta l'estensione ABPS sia per RTP che per SIP.
- il percorso tra il proxy server ABPS e il CN, basata nuovamente su RTP.

La parte critica del percorso appena elencato è quella tra client ABPS e proxy server ABPS; in quanto posso essere reti wireless dove la configurazione può cambiare a causa della natura mobile dell'utente. Il Policies sub-layer utilizza le estensioni di SIP e RTP per implementare meccanismi di QoS, come la scelta dell'interfaccia per il pacchetto da inviare, meccanismi di rilevamento di perdita del pacchetto, ritrasmissione anticipata e load balancing sul canale wireless. Questo livello implementa dei canali virtuali di QoS e li identifica come una coppia di porte SIP e RTP all'applicazione sul nodo mobile. Dunque, ogni applicazione del MN può scegliere la sua politica di QoS preferita semplicemente selezionando la porta SIP o RTP corrispondente. Questo permette ad ogni applicazione basata su SIP di inter-operare con l'architettura ABPS senza apportare alcuna modifiche.

3.2.1.5 Update

Una possibile fase di update può verificarsi quando vengono rilevati cambi della configurazione al MN. Il modello ABPS evita di utilizzare i messaggi re-INVITE in quanto le estensioni introdotte a RTP includono nel datagram SRTP stesso le informazioni necessarie per continuare la comunicazione dopo che il mittente ha cambiato l'indirizzo IP. Quindi se una rete non è più disponibile, ABPS permette di utilizzarne automaticamente un'altra; l'Authentication Tag insieme alla porta UDP per i messaggi SRTP, identificano univocamente il flusso di dati RTP tra il MN e il CN. Questo approccio permette al proxy server ABPS di identificare il mittente non tenendo in considerazione l'indirizzo IP, l'interfaccia o la rete che il mittente sta utilizzando per trasmettere il datagram. Una volta

che il mittente è stato identificato, il proxy server ABPS rileva il suo nuovo indirizzo IP dal campo "source IP address" del pacchetto IP.

3.3 ABPS nel nodo mobile

La struttura dell'architettura ABPS necessaria sul nodo mobile è composta da 4 principali componenti descritte nel dettaglio nei paragrafi 3.3.1, 3.3.2, 3.3.3 e 3.3.4.

3.3.1 Monitor

Il Monitor è responsabile di monitorare e configurare le interfacce wireless, le regole di routing necessarie per il packet forwarding ed effettuare il reporting delle interfacce che sono attive. Ogni qual volta che un'interfaccia di rete è, o configurata o disabilitata come conseguenza di un errore di comunicazione, viene inviata una notifica al proxy client (Reconfiguration Notification). Il Monitor fa affidamento su alcune API che il sistema operativo Linux mette a disposizione. Sia per le interfacce UMTS che WiFi, il monitor configura delle regole di routing utilizzando i comandi "ip route" e "ip rule"; questi comandi impongono l'utilizzo di una determinata interfaccia per un determinato indirizzo IP sorgente; questo permette di scegliere per quale interfaccia trasmettere il pacchetto.

3.3.2 TED

Il Transmission Error Detector (TED) opera tra il livello MAC, rete, e trasporto del kernel Linux. La sua principale responsabilità è di monitorare ogni singolo datagram UDP inviato attraverso un'interfaccia WiFI per poter rilevare se questo è stato ricevuto con successo dall'access point o meno. TED notifica alla componente proxy client dei datagram che sono stati scartati. Per poter fare ciò, TED abilita ed estende la coda dei messaggi di errore del socket UDP, che il proxy client utilizza per trasmettere i messaggi SIP e RTP. Questa coda è

tradizionalmente utilizzata per consegnare a livello applicazione i messaggi ICMP ricevuti a livello di rete.

In primo luogo in ABPS è stata estesa la system call sendmsg, che trasmette un datagram UDP attraverso il socket UDP; questa chiamata consegna il datagram al sistema locale di buffer per quel determinato socket UDP; dunque il modulo di routing decide attraverso quale interfaccia effettuare la trasmissione per poi ritornare al chiamante. Il valore di ritorno è un parametro intero che ha il valore del campo id dell'header IP che precede il datagram UDP; questo valore è un identificatore unico temporaneo per quel datagram IP. Il proxy client usa la funzione sendmsg e mantiene l'id di ogni datagram inviato, aspettando per una notifica di trasmissione effettuata o non effettuata da livello MAC; la notifica contiene l'id del datagram IP che l'ha generata. Secondariamente, è introdotto una nuova notifica di errore, il messaggio IP_NOTIFY, che contiene l'id del pacchetto IP che lo ha generato.

Il proxy client utilizza questo sitema di notifiche esteso, abilita l'utilizzo della coda di messaggi per le socket, invia i datagram UDP utilizzando la system call sendmsg, memorizza gli id dei datagram IP che hanno incapsulato i datagram UDP inviati ed aspetta per la notifica IP_NOTIFY. Questi messaggi sono letti usando la system call recvmsg con il flag MSG_ERRQUEUE per poter capire quando un dato pacchetto UDP è stato inviato o scartato, e decidere se quel datagram deve essere ritrasmesso.

3.3.3 Proxy Client

A livello sessione il proxy client implementa il supporto per i messaggi SIP e RTP/RTCP e le firme digitali necessarie come descritto in 3.2.1. Il proxy client include un semplice modulo Policies che fornisce all'applicazione VoIP la necessaria interattività e una bassa perdita di pacchetti, grazie alle notifiche

ricevute dalle differenti componenti. Il proxy client utilizza i socket UDP per ogni interfaccia wireless attiva sull'host; per questi riceve tre tipi di notifiche:

- il primo tipo di notifiche arriva dalla componente Monitor, che informa il proxy client che una nuova interfaccia è stata configurata o che è stata disabilitata; per ognuna di queste nuove interfacce configurate, il proxy client registra un socket UDP. Al contrario, per ogni interfaccia disabilitata il proxy client chiude il socket UDP corrispettivo, e valuta come persi i pacchetti che aveva inviato attraverso quel socket per cui non sono state ricevute notifiche TED "First-hop Transmission Notification".

- il secondo tipo di notifiche arriva dal protocollo ICMP che può notificare quando il proxy server è irraggiungibile da quella determinata interfaccia.

- il terzo tipo di notifiche è generata da TED che informa il proxy client che un datagram UDP è stato ricevuto o scartato dall'access point WiFi. Se non arrivano notifiche TED al proxy client entro 30 msec dopo che il datagram è stato inviato, il proxy client assume che il datagram sia stato perso.

Basandosi su questi tre tipi di notfiche, il proxy client decide se un datagram UDP deve essere ritrasmesso utilizzando un'altra interfaccia wireless tra quelle disponibili o deve essere scartato permanentemente; inoltre basandosi su queste notifiche il proxy client può selezionare l'interfaccia da utilizzare per inviare il prossimo datagram UDP.

3.3.4 Proxy DNS

L'ultima componente è il proxy DNS che riceve richieste DNS dalle applicazioni locali ed effettua richieste multiple ai server DNS in parallelo tramite tutte le interfacce disponibili. La prima risposta ricevuta dal server DNS è trasmessa

all'applicazione locale che ha effettuato la richiesta. L'obiettivo di questa componente è di evitare che una richiesta DNS non riceva una risposta in tempo a causa del fault dell'unica interfaccia attraverso la quale è stata trasmessa. Il proxy DNS è implementato estendendo un DNS proxy open source, dnsmasq.

Le componenti descritte sono rappresentate in Figura 4, dove partendo dai livello dello stack ISO/OSI, si può osservare come sono distribuite all'interno dell'implementazione del nodo mobile. La parte centrale mostra qual'è l'interazione riguardante lo scambio di pacchetti (evidenziandone anche il verso); colonna a destra invece prende in considerazione le componenti dell'architettura del MN che riguardano la configurazione delle interfacce di rete.

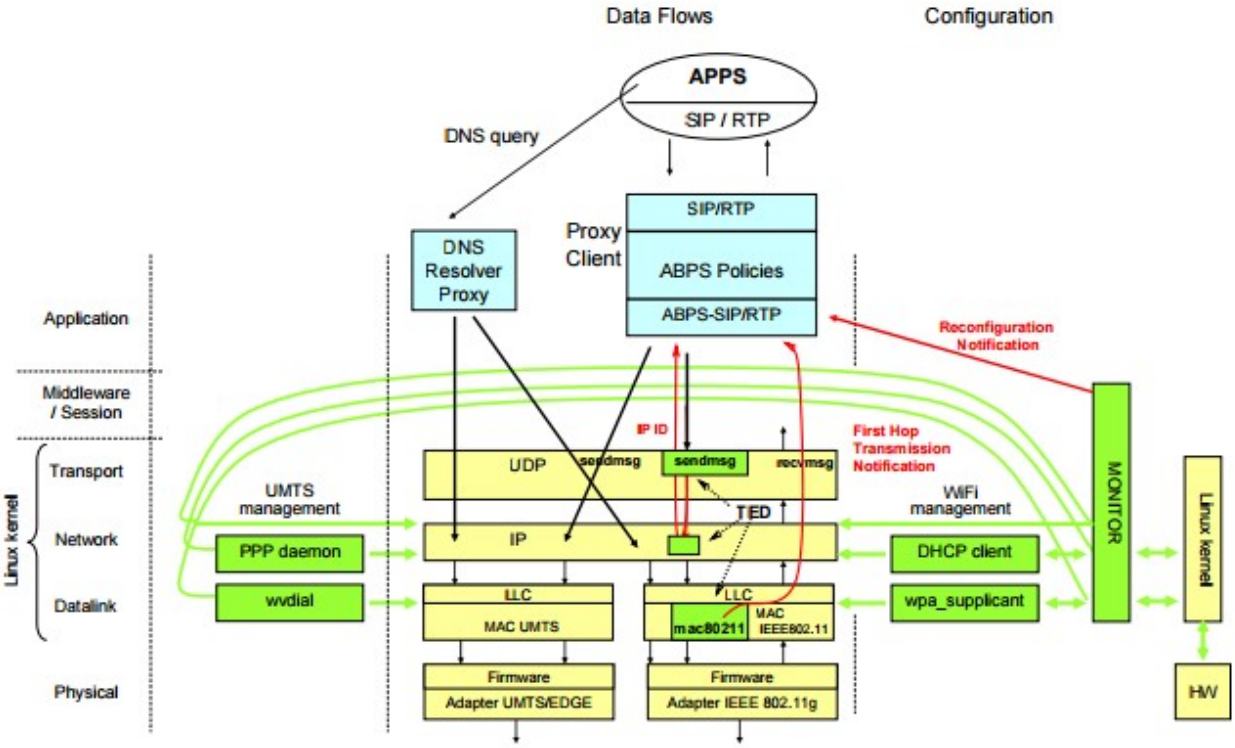


Figura 4 - Implementazione sul nodo mobile.

Capitolo 4

Simulatori

4.1 Introduzione

Per poter testare le prestazioni delle due architetture per il supporto alla mobilità trattate, si è scelto di utilizzare il simulatore ad eventi discreti OMNeT++ ed il suo framework INET. Questa decisione è stata intrapresa in quanto era già stato implementato in parte su INET, da altri ragazzi dell'Università di Bologna, il protocollo ABPS; da questi sono stati affrontati ed inseriti maggiormente gli aspetti riguardanti il proxy client sul nodo mobile: ritrasmissione anticipata, sistema di notifiche, scelta ed utilizzo di più interfacce in modo trasparente ad una sola applicazione. Come spiegheremo più avanti è stato necessario implementare alcune nuove funzionalità legate al proxy server ABPS, e modificare altre componenti per permetterne un più corretto funzionamento. Il simulatore INET per LISP è stato implementato nel 2012 da alcuni membri dell'Università di Wuerzburg, che hanno poi pubblicato l'articolo riguardante la stessa l'integrazione di LISP e LISP-MN nel framework INET di OMNeT++ [6]; questo copre gran parte dell'architettura LISP, l'unica componente tralasciata è il sistema LISP+ALT, di cui viene spiegato il funzionamento precedentemente nel paragrafo 2.3.3 e per approfondimenti [4], che viene però idealmente approssimata dal simulatore. In questo capitolo dopo aver affrontato i dettagli riguardanti i due simulatori utilizzati, verranno elencate e spiegate le modifiche apportate da ambo i lati.

4.2 LISP e LISP-MN per INET

L'implementazione dell'architettura LISP è stata descritta ed esaminata in 2.

Questo simulatore presenta diverse componenti che modellano i comportamenti di questa architettura; partendo dalle entità di tunneling ITR e ETR, che sono per comodità presi in considerazione come unico nodo LispRouter, sino a Map Resolver e Map Server: il primo inserito all'interno di LispRouter, il secondo lasciato come componente esterna; il Map Server è l'interfaccia globale per un possibile sistema di mapping distribuito, non implementato però dal simulatore. Di seguito vengono elencate le componenti principali che compongono lo strumento e le loro principali funzionalità:

- LispRouter

  È l'elemento che rappresenta il router LISP generico, che esso sia un ITR o un ETR. In questo simulatore il nodo di rete ricopre entrambe le funzionalità, e può inoltre essere impostato come Proxy ITR per un sito non-LISP. Il PITR dovrebbe effettuare routing advertising per l'instradamento dei pacchetti inviati a indirizzi EID dai siti non-LISP; in questa implementazione invece, è necessario configurare staticamente la rete (tramite tabelle di routing irt)in modo che nelle reti non-LISP, i pacchetti che come destinatari avessero un EID vengano instradati verso il Proxy ITR.

- Lisp-MN

  Questa componente rappresenta il nodo mobile nell'architettura LISP. Ne implementa tutte le funzionalità, anche il multihoming. Esso infatti si comporta come se fosse un piccolo sito LISP (incapsula e decapsula come se fosse l'ITR e l'ETR del proprio sito) in movimento; gli indirizzi IP che ottiene nelle reti dove fa roaming sono gli RLOC per il suo sito (che a loro volta sono EID). Questo gli permette di utilizzare in modo concorrente le proprie interfacce, non modificando il proprio EID globale; gli è consentito quindi mantenere la comunicazione con il CN anche quando il set dei suoi RLOC viene modificato.

- MapResolver

  In questo simulatore è una componente interna al RouterLisp, ed è l'interfaccia con il quale i vari nodi LISP possono richiedere il mapping per un determinato EID. Questi riconoscono un determinato dominio di prefissi EID, che deve essere configurato manualmente. In caso contrario viene utilizzato il domino di default (132.187.0.0/16). Se i pacchetti passati al modulo LISP espongono degli indirizzi IP che fanno parte del dominio degli EID, questa componente invia una Map-Request per quel determinato EID (nel caso non avesse il mapping già nella cache).

- MapServer

  Questa è la componente più idealizzata in relazione al suo effettivo ruolo all'interno del protocollo. Nel simulatore ne viene configurato solo uno, come se avesse il ruolo di database globale centralizzato; che è solo una semplificazione del protocollo, il quale vedrebbe un Map Server configurato nei pressi di ogni sito LISP, se non addirittura all'interno dello stesso nodo dell'ETR; come descritto in [3]. Per far sì che questa semplificazione non comprometta le prestazioni del protocollo, si può (come sopra suggerito) inserirne solamente uno che si comporti da database globale e impostare un tempo di processamento della Map-Request; questo tempo rappresenta il processamento della richiesta da parte del sistema di mapping distribuito utilizzato per questo scopo (ad esempio LISP+ALT). Oppure, come è stato impostato per questo lavoro, è possibile impostare dei link ad alta latenza, con lo scopo di introdurre un ritardo nella risposta da parte del database di mapping. Questa è inoltre la componente che modella il comportamento di un Map Server generico; dunque risponde e processa tutte i messaggi di signaling LISP riguardanti la registrazione (Map Register), la richiesta (Map Request) ed

inoltra come risposta rispettivamente, Map Notify e Map Reply.

- LispRouting

  È il modulo che effettua il routing basandosi sulla risoluzione dei mapping EID-to-RLOC; se non trova delle corrispondenze, o gli indirizzi presenti nel pacchetto non fanno parte del dominio degli EID configurato, lascia la gestione del routing al modulo nativo di INET per il livello di rete.

4.3 ABPS per INET

L'implementazione di ABPS per il framework INET consiste in una serie di modifiche, apportate al framework stesso, per poter simulare il comportamento del suddetto protocollo per il supporto alla mobilità. Nel simulatore che abbiamo trattato durante il nostro lavoro, le modifiche sono state effettuate per ogni livello della stack protocollare ISO/OSI; inoltre sono state introdotte diverse nuove componenti, tra le quali le più rilevanti sono elencate in seguito:

- AdvancedWirelessHostRWMA

  Questo modulo composto rappresenta l'host wireless generico, al quale possono essere configurati anche differenti modelli di mobilità; come nel nostro caso. Il modulo può inoltre impostare diverse applicazioni sia TCP che UDP con la restrizione che queste siano dello stesso tipo per il protocollo a livello trasporto scelto. Il modulo presenta un vettore di interfacce wireless che ne permettono la comunicazione durante gli spostamenti con l'access point al quale sono allacciate.

- SimpleHostRWMA

  Questo modulo composto rappresenta l'host generico, ed è per molti aspetti simile a quello precedentemente descritto; ad eccezione del modulo riguardante la mobilità, che qui non è possibile impostare, e ad

eccezione delle interfacce wireless che questo nodo non prevede. Infatti questo modulo è stato pensato per il collegamento "wired" alle altre componenti della rete, tramite ppp o ethernet.

I nomi delle due componenti sono caratterizzate dall'acronimo RWMA (Robust Wireless Medium Access) che rappresenta le politiche di notifiche, ritrasmissione anticipata e scelta ad hoc dell'interfaccia per ogni pacchetto applicate dal client proxy ABPS.

Per ogni livello della stack protocollare in questo simulatore sono presenti alcune componenti che modellano il comportamento di RWMA ed in parte anche di ABPS; in Tabella 1 sono elencate (a seconda della loro posizione nello stack ISO/OSI) solo quelle più vicine all'analisi svolta, che riguarda la comunicazione su UDP; inoltre, il simulatore presenta modifiche apportate a diverse altre componenti di INET, che però non vengono trattate in questo lavoro di tesi.

Applicazione	Trasporto	Rete	Datalink	Fisico
ULBRWMA	UDPRWMA	IPRWMA	Ieee80211NicSTARWMA	NullMobilityObstacle
ULBRWMAServerProxy		ARPRWMA	Ieee80211AgentSTARWMA	
UDPBasicAppForMultipleNics			Ieee80211MgmtSTARWMA	
UDPBasicAppForServerProxy			Ieee80211MgmtAPRWMA	
			Ieee80211MacRWMA	
			Ieee80211MacApRWMA	

Tabella 1

Partendo dalle componenti elencate in tabella 1, sono state prese in considerazione solo alcune di queste, perlopiù i moduli a livello applicazione e gli ostacoli a livello fisico; come verrà descritto 4.4.3 è stato necessario estendere le funzionalità di ABPS già implementate per poter modellare anche

il comportamento dell'ultimo step della comunicazione di ABPS: quello tra proxy server ABPS e CN. Questa implementazione, per simulare il server proxy utilizza il modulo SimpleHostRWMA, sul quale è necessario configurare un'applicazione UDP di tipo UDPBasicAppForServerProxy, che estende ULBRWMAServerProxy. In 4.3.1 e 4.3.2 verranno esposti nel dettaglio il comportamento del proxy client e del proxy server, modellati dai loro rispettivi moduli a livello applicazione.

4.3.1 Proxy Client ABPS

Il proxy client è definito in questo simulatore, per quanto riguarda il livello applicazione, da ULBRWMA e UDPBasicAppForMultipleNics. La prima è la componente che modella il proxy client ABPS, mentre la seconda è un'applicazione UDP generica che viene interfacciata con il proxy client. Le due componenti potrebbero essere separate per maggiore chiarezza di modellazione, ma sono state implementate una (UDPBasicAppForMultipleNics) l'estensione dell'altra (ULBRWMA). Di seguito vengono descritte le loro funzionalità:

- ULBRWMA

  Vengono introdotti in questa classe tutte le caratteristiche e le funzionalità legate alla gestione delle notifiche di errore e di riconfigurazione, legate all'approccio descritto in 3.3.3. Le interfacce sono configurate in relazione alle notifiche di riconfigurazione ricevute. Questa componente presenta anche dei comportamenti legati al client e al server DHCP, e grazie a questi è in grado di iniziare la procedura di richiesta dell'indirizzo IP. Le funzionalità di DHCP ad ogni modo sono implementate in modo piuttosto rigido, richiedendo un indirizzo predefinito per il server DHCP. Questa è una grossa limitazione per l'impostazione di scenari simulativi elaborati, ed è inoltre poco verosimile in quanto per rintracciare un DHCP in una rete esiste una determinata

procedura propria del protocollo.

- UDPBasicAppForMultipleNics

  Questo modulo è quello che introduce l'invio vero e proprio di pacchetti UDP verso la destinazione; inoltre sono implementate anche funzionalità di collezione statistica al suo interno. I pacchetti vengono inviati ad un indirizzo di destinazione che deve essere il proxy server ABPS o un altro host al quale questi pacchetti devono essere recapitati.

4.3.2 Proxy Server ABPS

Il proxy server è analogamente definito come il client; i suoi comportamenti sono modellati da due componenti con gli stessi legami di ereditarietà, ULBRWMAServerProxy e UDPBasicAppForServerProxy. Le funzionalità introdotte da questo modulo sono quella di proxy server e di CN, che oltre a ricevere i pacchetti dal nodo mobile potrà inviarne di rimando.
Concettualmente è stato definito così per rappresentare il caso peggiore per l'architettura ABPS: il server proxy posizionato nel punto più distante dal nodo mobile, introducendo un maggior ritardo nella fase di update una volta che questo avrà cambiato il proprio indirizzo IP. Di seguito vengono esposte le varie funzionalità delle due classi:

- ULBRWMAServerProxy

  È l'implementazione del proxy server vero e proprio. Riceve i pacchetti UDP dal MN e ne associa il proxy client id all'indirizzo IP del mittente, analogamente al reale comportamento di ABPS. Non è stata inserita però alcun tipo di meccanica che effettui l'inoltro dei pacchetti attraverso l'ultimo step del percorso del percorso di ABPS. Se si volesse realmente fare sperimentazioni posizionando il proxy server nella backbone o comunque distante dal CN non si potrebbe, in quanto la funzionalità stessa che re-inoltra il pacchetto verso la destinazione non è presente in

questa implementazione di ABPS.

- UDPBasicAppForServerProxy

  Questa componente rappresenta il vero e proprio CN che, in questo caso, risiede nel proxy server. Può essere configurata come ogni altra applicazione UDP di INET con l'unica differenza che l'indirizzo IP di destinazione dei pacchetti UDP viene ogni volta scelto casualmente tra i vari indirizzi IP registrati nella mappa *<proxy client id : ip address>*. Questo, in presenza di più nodi mobili, può eventualmente risultare in comportamenti inappropriati del CN, che si troverebbe ad inviare ogni pacchetto scegliendo casualmente come destinazione uno degli host registrati al proxy server.

4.4 Modifiche effettuate

4.4.1 Calcolo Penetration Loss

La parte del simulatore ABPS, che poi è stata importata anche in LISP per il calcolo della penetration loss indotta dagli ostacoli che ostruiscono il segnale radio, presentava un calcolo di questa errato di seguito riportato:

```
for( per ogni ostacolo che ostruisce il segnale )
totalPenetrationLoss += FWMath::dBm2mW((*obstacleIt)->getPenetrationLoss());

rcvdPower *= pow(M_E, (-totalPenetrationLoss * distance));
```

Analizzando le seguenti linee (dove totalPenetrationLoss rappresenta la perdita di segnale introdotta dagli ostacoli e rcvdPower la potenza di segnale ricevuta a destinatario), si evince che la penetration loss calcolata è una somma di tutti i valori mW delle ostruzioni al segnale; dunque se abbiamo 2 muri sul tragitto del sengale ed una perdita di -3dBm per ogni muro, che sono circa 0.5 mW

rispetto all'unità, otteniamo (come calcolato sopra) una penetration loss totale di 1 mW. Un calcolo più corretto si otterrebbe sommando i valori in dBm delle varie ostruzioni, ottenendo come risultato -6dBm, che sono ~0.25 mW rispetto all'unità. Inoltre la formula con la quale si calcola la potenza del segnale al ricevitore è poco chiara ed apparentemente scorretta; si è optato dunque per calcolare la penetration loss e la potenza del segnale a destinatario come segue, convertendo la somma dei dBm di *penetrationLoss* in mW e moltiplicandoli per la potenza del segnale al ricevitore:

```
for( per ogni ostacolo che ostruisce il segnale )
totalPenetrationLoss += (*obstacleIt)->getPenetrationLoss();

totalPenetrationLoss = FWMath::dBm2mW(totalPenetrationLoss);

rcvdPower *= totalPenetrationLoss;
```

4.4.2 Modifiche effettuate su LISP

4.4.2.1 Definizione Proxy-ITR

Nella versione del simulatore LISP scaricato, implementato da Michael Hoefling e Dominik Klein, erano presenti delle rigidità per quanto riguarda la configurazione degli scenari simulativi. La più notevole tra queste è la persistente necessità della presenza di un modulo LispRouter che abbia nome PITR (Proxy-ITR), il quale se non è presente durante l'inizializzazione lancia un errore. In uno scenario reale, riguardo l'aspetto protocollare, LISP ha bisogno della presenza di un Proxy-ITR almeno per sito non-LISP, quindi quella di tenere in considerazione un unico modulo, identificato da un unico nome, è decisamente una restrizione importante in vista dello studio di scenari che possano modellare verosimilmente questo comportamento del protocollo LISP.

Si è deciso dunque di permettere di specificare il nome del modulo da

parametro per ogni Map-Resolver. Nonostante questo, per permettere che il simulatore modelli il comportamento della comunicazione intra-site tra non-LISP e LISP in linea con le procedure protocollari specificate in [5], questa modifica non è sufficiente; il parametro specificato servirebbe, come specifica l'implementazione di Hoefling e Klein, a definire il destinatario per l'invio delle varie SMR e Piggybacked Map-Request da parte di chi subisce un aggiornamento delle proprie mapping entry (il nodo mobile); viene tenuto in considerazione il caso in cui il nodo mobile si sposta da una rete all'altra, inviando una nuova Map-Register e ricevendo una Map-Notify, che fa scattare le procedure di SMR o Piggybacked Map-Request; questo comportamento può essere definito una semplificazione del Temporary PITR Caching suggerito per l'aggiornamento di cache remote in [2]. Quindi in futuro, se si dovesse pensare a scenari più elaborati, si dovrebbe eliminare completamente questo modo di gestire l'aggiornamento delle mapping cache dei PITR, ed implementare una procedura più coerente a quella specificata in [2], che prevede una cache di PITR per i quali il nodo mobile ha ricevuto Map-Request, ai quali inoltrerà la SMR una volta che le mapping entry subiranno dei cambiamenti.

4.4.2.2 Definizione Proxy-ETR

Una configurazione simile era implementata per la gestione del Proxy-ETR, il quale indirizzo veniva specificato da codice in maniera statica; per ogni simulazione infatti, secondo l'implementazione di Hoefling e Klein, il PETR è sempre il nodo con indirizzo 192.168.0.6. Questa, come per la definizione del PITR precedentemente, è una restrizione particolarmente rigida in quanto oltre a non permettere di impostare in modo parametrico il modulo da utilizzare come PETR, non permette di definite un Proxy-ETR diverso per sito LISP, come consigliato in [5]. In questo caso come per il precedente, è stato aggiunto un parametro al modulo LISPRouting con nome PETRAddress; questo è un indirizzo IPv4 che permette al singolo nodo LISP di definire il proprio Proxy-ETR

come specificato in [5], al quale invierà i pacchetti incapsulati LISP per l'inoltro a siti non-LISP.

4.4.2.3 Applicazione UDPLISPApp

Per permettere una configurazione delle comunicazioni tra due nodi tramite LISP più immediata, per diverse tipologie di scenari e reti, si è voluto inserire un modulo applicazione di tipo UDP ad hoc, UDPLISPApp. È un estensione dell'applicazione UDP di base che, a differenza di quest'ultima, permette un'impostazione del nodo destinatario dei pacchetti semplificata; tramite il parametro destinationNode, il nome del modulo destinatario, viene poi estratto l'indirizzo IP automaticamente durante l'inizializzazione della comunicazione UDP: l'indirizzo IP viene estratto relativamente al tipo di nodo destinatario, nel caso si tratti di un nodo statico, viene ottenuto il suo indirizzo IPv4 per la rete nella quale si trova (che può essere considerato come un EID), altrimenti viene utlizzato il parametro *mobileEid* per estrarne l'EID LISP, assumendo che il modulo faccia riferimento ad un nodo mobile. È stato inoltre inserito il parametro *startTime*, che permette di specificare un tempo nel quale l'applicazione UDP inizia ad inviare i pacchetti. Questo modulo assume che al momento dell'invio del primo pacchetto dopo l'intervallo startTime, il nodo destinatario abbia terminato la configurazione del proprio indirizzo IP, in quanto dovrà essere utilizzato come indirizzo destinatario per i pacchetti UDP.

4.4.2.4 Altre modifiche

Oltre alle modifiche di maggior rilevanza elencate precedentemente, è stato necessario apportare alcune correzioni per evitare di incorrere in errori e per rimanere coerenti rispetto agli altri simulatori presi in considerazione.

La sensitivity nel modello radio veniva prima assegnata utilizzando come unità di misura i milliwatt, in contrasto con gli altri due simulatori che utilizzano i

decibel mW; si è dunque deciso di modificare l'unità di misura per la sensitivity anche per il simulatore LISP.

In LISP si verificava la situazione in cui il nodo mobile, una volta ricevuta la Map-Notify dal Map-Server dopo essersi registrato al database, si trovava a non poter inviare la SMR agli RLOC presenti nella propria Map-Cache a causa di mapping incompleti. Per ovviare a questo problema si è deciso di inviare una Map-Request per ogni mapping incompleto dopo la Map-Notify.

4.4.2.5 Porting degli ostacoli

Il porting degli ostacoli dal simulatore di ABPS, dove già erano implementati, al simulatore LISP è stato relativamente poco complesso, complice la simile versione dei due framework INET utilizzati per i due simulatori. I file che implementano il comportamento degli ostacoli sono quelli elencati in Tabella 2 e sono stati direttamente importati nel simulatore LISP.

mobility	world
MobilityObstacle.cc MobilityObstacle.h MobilityObstacle.ned	NullMobilityObstacle.ned BasicObstacle.cc BasicObstacle.h BasicObstacle.ned

Tabella 2

Oltre alla copia dei file è stato necessario modificare altre componenti di INET per poter permettere agli ostacoli di funzionare correttamente; i file sopra descritti definiscono unicamente il comportamento di un singolo ostacolo di mobilità stazionaria (muro), ma non ne definiscono l'interazione con le onde radio utilizzate per la trasmissione dei dati. È dunque stata inserita la lista degli ostacoli presenti nello scenario in *ChannelControl*, la classe che rappresenta il

canale di comunicazione per le onde elettromagnetiche. Questa lista verrà controllata dal modulo radio dei nodi alla ricezione di un pacchetto; viene calcolata l'intersezione con la traiettoria della comunicazione per ogni ostacolo, e poi sommata la *penetrationLoss* (perdita del segnale dovuta all'ostacolo) di ognuno di questi attraversato. Viene infine calcolato il nuovo valore di potenza del segnale secondo la formula descritta in 4.4.1.

4.4.3 Modifiche effettuate su ABPS

4.4.3.1 Modifiche a livello datalink

Le modifiche a questo livello della pila protocollare riguardano due comportamenti, uno proprio della versione di INET utilizzata e l'altro è un aspetto introdotto dal tesista Luca Regazzi in [9]. La prima modifica è stata necessaria per permettere agli access point di inviare ad ogni stazione associata i messaggi di broadcast. Il primo comportamento anomalo era definito nel file che definisce le funzionalità del modulo che gestisce i management frame dell'access point, *Ieee80211MgmtAP.cc*: questo scartava i pacchetti il quale indirizzo MAC di destinazione non veniva riscontrato nella mappa delle stazioni associate a quel determinato access point, senza però verificare che questi potessero essere messaggi di broadcast a livello datalink. È stato aggiunto successivamente il controllo necessario per consentire ai messaggi broadcast di essere trasmessi; questi verranno poi utilizzati nella modifica della funzionalità di client e server DHCP nelle classi di ABPS a livello applicazione.

Il secondo comportamento anomalo a livello datalink era dato dalle funzionalità di scarto dei pacchetti lato nodo mobile e scarto dei pacchetti lato access point, introdotte da Luca Regazzi in [9], che servivano per produrre dei test e delle simulazioni più verosimili, ma che impedivano di poter confrontare in modo coerente l'implementazione di ABPS con il simulatore LISP; quindi abbiamo

dovuto eliminare questa funzionalità, rimuovendo le definizioni del preprocessore che la attivavano in *Ieee80211Mac.h*.

4.4.3.2 Modifiche a livello rete

A livello rete le modifiche effettuate sono due e riguardano il modulo IP che gestisce il routing e l'instradamento dei pacchetti, e il modulo ARP che si occupa della risoluzione degli indirizzi IP in MAC. Il file IP.cc non permette ad un pacchetto di avere un indirizzo sorgente che non appartenga ad una delle interfacce presenti nell'host; dunque non ci permette di implementare il comportamento di ABPS che vede il proxy server ABPS inoltrare i pacchetti verso il CN in vece del nodo mobile. Questa restrizione è stata dunque eliminata per consentire a livello applicazione l'implementazione del proxy server ABPS.

Per quanto riguarda ARP invece, la modifica richiesta è stata più complessa. È stato constatato che la presenza di una sola cache ARP può portare a due casi in cui, con le due interfacce connesse a due reti diverse ma utilizzate concorrentemente dalla stessa applicazione, il simulatore presenta delle anomalie nel comportamento:

Caso 1 - Risoluzione ARP condivisa

- L'interfaccia *A* e l'interfaccia *B* del nodo mobile sono collegate a due reti differenti.
- Il nodo vuole inoltrare un pacchetto tramite *A* verso il suo destinatario *X*.
- *A* ottiene il prossimo passo verso *X* tramite risoluzione ARP.
- *A* inserisce l'indirizzo IP per il next-hop in cache ARP.
- Per qualche motivo l'interfaccia *A* non è più utilizzabile e l'applicazione inoltra il prossimo pacchetto attraverso *B*.

- *B* utilizzando la stessa destinazione, al momento del controllo nella cache ARP (se questa non è ancora scaduta) effettua un cache-hit ed invia il messaggio al next-hop ottenuto. Questo è il comportamento che deve essere modificato per evitare la successiva perdita di pacchetti.
- Il messaggio viene scartato in quanto l'indirizzo al quale è stato inviato si trova in una rete differente.

Caso 2 - ARP Reply errata

- L'interfaccia *A* del nodo mobile è collegata ad una rete mentre l'interfaccia *B* non è configurata.
- Il CN si trova nella stessa rete alla quale è collegata l'interfaccia *A* del nodo mobile; con la quale assumiamo stia comunicando.
- Il nodo mobile configura con successo l'interfaccia B, che si allaccia ad una nuova rete, iniziando la comunicazione su quella: l'indirizzo per quel proxy-client-id al proxy server viene aggiornato.
- Il CN che vuole inviare un pacchetto al nuovo indirizzo inizia una nuova risoluzione ARP.
- L'ARP Request arriva ad un router che conosce come raggiungere la destinazione; questo di rimando invia la ARP Reply al CN, il quale, una volta ricevuta, aggiornerà la sua cache ARP.
- L'ARP Request arriva anche al nodo mobile che ha l'interfaccia *A* ancora collegata alla rete condivisa con il CN.
- L'interfaccia *A* risponde alla ARP Request con una ARP Reply per l'indirizzo dell'interfaccia *B,* che andrà a sovrascrivere la risposta fornita precedentemente dal router. Da implementazione di INET è sufficiente che l'indirizzo appartenga ad una delle interfacce dell'host perchè questo invii la risposta. Questo è l'aspetto che verrà corretto per quanto riguarda questo problema.

- Il CN invia i pacchetti seguenti all'indirizzo MAC di *A* per l'indirizzo IP di destinazione di *B.*
- Una volta che *A* non sarà più connesso alla rete del CN, tutti i successivi pacchetti verranno persi finche non verrà effettuata una nuova risoluzione ARP.

Per risolvere questi due problemi legati ad ARP è stato implementato un comportamento che simula la presenza di una cache ARP per ogni interfaccia di rete. Infatti, in INET, la classe ARP già memorizza l'interfaccia per la quale è stata creata la entry nella cache ARP, che è rappresentata da una mappa *<ip address : cache entry>.* La chiave della mappa è l'indirizzo IP per il quale si è avviata la procedura di risoluzione ARP, mentre come valore si trovano tutte le informazioni riguardanti la entry, tra i quali il puntatore all'interfaccia attraverso la quale è stata inoltrata la richiesta di risoluzione ARP.

Si è dunque implementato un controllo aggiuntivo nel momento in cui si cerca un riscontro nella cache ARP, imponendo che l'interfaccia per la quale si è creata la entry sia la stessa che vuole inoltrare il pacchetto. Così facendo, anche se è già presente una entry nella cache per un determinato indirizzo di destinazione, questa non verrà ritenuta valida a meno che non sia stata realmente ottenuta tramite l'interfaccia che si sta utilizzando.

È stato inoltre aggiunto un controllo al momento della ricezione di una Map Request, dove viene verificato che l'indirizzo per il quale si sta rispondendo sia effettivamente quello che appartiene alla nostra interfaccia. Questo comportamento ovviamente è implementato come estensione del normale modulo ARP, che viene esteso in ARPRWMA, e da noi successivamente modificato; i router e le altre componenti della rete continuano ad utilizzare il normale modulo implementato nativo in INET.

4.4.3.3 Modifiche a livello applicazione

Il livello applicazione è stato quello dove sono state introdotte le modifiche più drastiche; queste con lo scopo di inserire nella simulazione del protocollo ABPS anche l'ultimo passo del suo percorso: quello tra proxy server e CN. Come spiegato anche in 4.3.2, il modulo che aveva il compito di modellare il proxy server ABPS, in questa implementazione comprendeva anche i comportamenti propri del correspondent node. Non era possibile inoltre, con i moduli a disposizione, configurare uno scenario dove un numero di applicazioni o nodi mobili *N* comunica con un numero di applicazioni o nodi destinatari *M* attraverso il proxy server ABPS (o diversi proxy server ABPS). Questo a causa delle restrizioni di locazione che vedono il CN nello stesso nodo del proxy server, e impongono all'applicazione del nodo mobile un indirizzo IP come destinatario della comunicazione UDP; sarebbe più appropriato instaurare la comunicazione attraverso un proxy client id.

Le modifiche apportate sono diverse, e vanno ad incidere su ogni componente del livello applicazione. Di seguito un elenco:

- ABPSUDPApp

  È stata inserita, sul modello di UDPBasicAppForMultipleNics, una nuova applicazione con il nome ABPSUDPApp; questa cerca di modellare in maniera più vicina alle specifiche di ABPS, definite in [7] il comportamento e la configurazione dell'applicazione del nodo mobile. L'applicazione ora incapsula ogni pacchetto e lo invia al proxy server specificato parametricamente da *proxyServerAddress,* l'indirizzo IP del proxy server ABPS. Al pacchetto, prima che esso venga inviato, vengono aggiunte informazioni riguardanti il proxy client id del destinatario e la porta sulla quale comunicare i dati UDP. Il pacchetto viene dunque inviato utilizzando la funzione *sendPacket* definita in *ULB.cc;* lasciando al

load balancer la decisione dell'interfaccia di invio.

Per configurare correttamente un'applicazione UDP che utilizzi l'architettura ABPS è dunque necessario configurare i seguenti (nuovi) parametri:

- *receiverId* - indice numerico intero che identifica il proxy client id di destinazione.
- *clientId* - indice numerico che identifica il proxy client id di questa applicazione.
- *destPort* - numero intero identificativo della porta sulla quale il proxy client al destinatario riceverà il messaggio.
- *localPort* - numero intero identificativo della porta sulla quale il proxy client riceverà la comunicazione da altre applicazioni UDP.
- *proxyServerAddress* - indirizzo IP (string) che corrisponde all'indirizzo IP del proxy server ABPS che si vuole utilizzare come outbound server per la comunicazione.
- *proxyServerPort default(590)* - numero intero identificativo della porta sulla quale si invieranno i paccheti incapsulati per il proxy server ABPS.

Questi nuovi parametri permettono all'applicazione di poter impostare un canale di comunicazione multi-path verso il destinatario, in quanto il proxy server si occuperà di inoltrare il traffico verso il CN a seconda del mapping che viene fornito per il proxy client id segnalato come destinatario nel pacchetto ricevuto. Inoltre questa versione dell'applicazione è implementata in maniera tale che: il primo pacchetto inviato dalla stessa non sia un pacchetto VoIP ma bensì la ABPS REGISTER; questa avrà come destinatario il proxy server, e serve per registrare il proprio proxy client id ad esso senza essere inoltrato a destinatario, prima dell'inizio della comunicazione vera e propria.

- ABPSProxyServerApp

  Questa classe, partendo dalla precedente *ULBRWMAServerProxy*, è stata ripulita dalle funzionalità di ULB, per lasciare solamente quelle relative alla gestione dei proxy client id e del DHCP. Inoltre, così come *ULBRWMAServerProxy* era implementata, non è presente la funzionalità di inoltro verso i relativi destinatari della comunicazione, come vorrebbe da specifica il protocollo.
  Dunque, partendo dalla mappa <*proxy client id : ip address*> si è sviluppata la funzione *decapsulateAndForward* che effettua, dato un proxy client id di destinazione (specificato nel pacchetto), il decapsulamento e l'inoltro verso l'indirizzo IP destinatario ottenuto dal mapping, inserendo come mittente l'indirizzo IP dell'interfaccia utilizzata dal MN per la comunicazione. Questo modulo nel caso riceva una ABPS REGISTER, inserirà la relativa coppia proxy client id e indirizzo IP sorgente nella sua mappa. L'unico parametro di questo modulo è *localPort*, di default definita a 590, è la porta sulla quale i proxy client devono mandare i loro messaggi per utilizzare il proxy server ABPS.

- ULB

  Questa classe è stata creata partendo dalla precedente *ULBRWMA;* viene estesa da *ABPSUDPApp* per modellare i comportamenti di un'applicazione VoIP che utilizzia il servizio di proxy client ABPS. Di questa sono state modificate perlopiù le caratteristiche legate alle sue funzionalità di client e server DHCP. Precedentemente era imposto di specificare l'indirizzo IP statico al quale contattare il server DHCP per poter configurare una delle proprie interfacce; limitazione che abbiamo voluto eliminare, semplificando la configurazione degli scenari ed aumentandone la chiarezza. Prima di procedere a spiegare le modifiche effettuate su

questa componente, bisogna precisare che, negli scenari in cui si vuole simulare l'architettura ABPS, è necessario che i server DHCP abbiano a livello applicazione *ABPSUDPApp* con parametro *clientDHCP* impostato a false. Questa scelta è legata all'implementazione di *UDPBasicAppForMultipleNics*, che era utilizzata sia come server DHCP che come client DHCP: è consigliabile in futuro utilizzare un DHCP preesistente, come quello che fornisce INET; che comporta però la necessità di modificare successivamente *ABSUDPApp* per renderla compatibile con il DHCP di INET.
Come già anticipato è stato aggiunto un parametro, clientDHCP; questo specifica se l'applicazione si comporta come server DHCP o come client DHCP. Questo ci ha permesso di impostare i messaggi DHCPDISCOVER e DHCPOFFER come messaggi di broadcast, eliminando la limitazione discussa in precedenza.

- VOWFPacket

  Al pacchetto UDP inviato dall'applicazione, sono stati aggiunti una manciata di parametri aggiuntivi; quelli necessari per un'implementazione basilare che sia in grado di simulare un comportamento simile a quello dell'architettura ABPS. I parametri inseriti sono i seguenti:
  - receiverId - numero intero che identifica il proxy client id destinatario del pacchetto.
  - receiverPort - numero intero che identifica la porta, sulla quale l'applicazione del destinatario si aspetta di ricevere i pacchetti.

Capitolo 5

Test

5.1 Introduzione

Una volta compreso il meccanimo e l'obbiettivo che ognuno dei simulatori propone come supporto alla mobilità, e dopo aver sistemato le funzionalità dei vari simulatori, almeno quelle che potevano produrre risultati ambigui o poco coerenti tra loro (es. errori di varia natura), sono stati effettuati diversi test per confrontarne l'efficienza in differenti scenari. I test effettuati comprendono i vari casi di comunicazione e intercomunicazione di LISP, la comunicazione con un Lisp-MN dotato, in alcuni casi, di più interfacce (multihomed) e vari casi di comunicazione dell'archiettura ABPS, dove il suo nodo mobile è sempre dotato di più di una sola interfaccia. Le metriche che si sono volute studiare nel confronto dei risultati, sono differenti tra loro, ma nell'insieme hanno lo scopo preciso di permettere la valutazione di un'architettura per il supporto alla mobilità nei suoi vari aspetti; queste metriche sono dunque le seguenti:

- *Packet delays* - questa è il tempo di percorrenza del pacchetto nella rete una volta che questo è stato inviato. Quindi permette di fare una stima sulla latenza media della comunicazione e di come questa si modifichi in relazione agli spostamenti del nodo mobile.

- *Sender Handover Delays* - questa è il periodo di tempo misurato al CN tra l'ultimo pacchetto ricevuto dal MN con un determinato indirizzo IP sorgente, e quello successivo con un indirizzo IP sorgente differente. Questa metrica può essere interpretata come intervallo di indisponibilità in invio da parte del MN (o in ricezione da parte del CN). È stato

necessario l'implementazione di questa metrica nel codice sorgente dei vari simulatori coinvolti, in quanto prende in considerazione un aspetto particolare della comunicazione che non è altrimenti possibile ottenere.

- *Receiver Handover Delays* - questa è il periodo di tempo misurato al MN tra l'ultimo pacchetto ricevuto dal CN con un determinato indirizzo IP destinatario, e quello successivo con un indirizzo IP destinatario differente. Questa metrica può essere interpretata come intervallo di indisponibilità in ricezione da parte del MN (o in invio da parte del CN). È stato necessario l'implementazione di questa metrica nel codice sorgente dei vari simulatori coinvolti, in quanto prende in considerazione un aspetto particolare della comunicazione che non è altrimenti possibile ottenere.

- *Packet Arrived/PacketSent* - questa è il rapporto tra il numero di pacchetti ricevuti da uno dei due nodi e quelli inviati dall'altro (o viceversa). Permette di misurare l'affidabilità di consegna per ognuno dei protocolli di supporto alla mobilità confrontati.

5.2 Ambienti simulativi

Le architetture sono state messe alla prova in due diversi tipi di ambienti; un'ambiente simmetrico e ideale, che ci permettesse di misurarne le prestazioni nello scenario più sgombro da interferenze e schematico possibile; l'altro invece vuole proporre uno scenario urbano che presenti ai vari protocolli diverse interferenze fisiche (ostacoli) al segnale radio. Le due configurazioni, sebbene la rete impostata per lo scenario urbano prevede un'apparato più elaborato e organico, rilevano la stessa latenza di percorrenza della backbone. Ognuno degli scenari come già anticipato è composto da due differenti tipologie di rete:

- Rete periferica - questa rete è composta da diversi host collegati al router tramite un hub; rappresenta qualsiasi rete a bassa latenza nella quale, la lantenza introdotta dalla sua percorrenza sia minima; nelle configurazioni proposte è impostata nell'ordine dei microsecondi.

- Rete virtuale di interconnessione (backbone) - questa rete composta da link virtuali, che vogliono astrarre il tragitto compiuto nella rete di interconnessione, la backbone. Ogni collegamento di questa rete è ad alta latenza, ed in ogni simulazione è dell'ordine di decine di millisecondi. Con questa componente dello scenario si vuole idealizzare la comunicazione tra due reti periferiche, che però si trovano a grande distanza l'una dall'altra. Nell'interfaccia grafica i link di questa rete sono di colore azzurro e tratteggiati, con latenza (30ms).

Per ognuno degli scenari, sono stati configurati i diversi casi possibili previsti per ognuno dei simulatori, in modo da poterne valutare la variazione anche in termini di prestazioni. Per quanto riguarda quelli in cui si può presentare una rete LISP, si sono studiati due possibili configurazioni:

- CN in sito LISP

  Questo caso prevede che il correspondent node si trovi all'interno di un sito LISP, e dunque sia identificato da un indirizzo IP che faccia parte del dominio degli EID per lo scenario. Si suppone che questo sia il caso ottimo per LISP, in quanto non vengono introdotti dei ritardi aggiuntivi a causa del convogliamento del traffico LISP verso il Proxy ETR, che non è detto si trovi in posizione ottimale.

- CN in sito non-LISP

  Questo caso prevede che il CN, con il quale il nodo mobile LISP vuole comunicare, si trovi sotto una rete non-LISP. Quindi come specificato in

[5] è necessario introdurre il Proxy ITR e il Proxy ETR (in caso di limitazioni da parte degli access network), che si occupano rispettivamente di incapsulare LISP verso gli RLOC destinatari e decapsulare LISP i pacchetti inviati dagli ITR per inoltrare verso i nodi che non fanno parte dei siti LISP. Non è comunque possibile configurare la rete in direzione delle specifiche definite in [5], a causa delle limitazioni imposte dal simulatore; nonostante ciò è possibile definire un unico PITR e un unico PETR per la rete, che sono sufficienti per gestire il traffico di pochi siti LISP, anche se la loro posizione non è comunque in linea con le specifiche.

I casi in cui è possibile differenziare le possibili configurazioni di ABPS, dove la variazione è definita dalla distanza del Proxy Server ABPS dal nodo mobile, sono le seguenti:

- Proxy Server nella rete del CN

  Questo caso si dovrebbe presentare come il peggiore per ABPS, in quanto, il tempo necessario al nodo mobile per effettuare la fase di update è il più alto che si possa ottenere tra le varie configurazioni; infatti, in questo caso la latenza del mesaggio è tutta causata dal percorso multi-path tra i due proxy, quello che vede il pacchetto essere inoltrato dal proxy client ABPS fino al proxy server ABPS. Si prevede quindi che questo sia il caso in cui gli intervalli di indisponibilità siano maggiori che per le altre configurazioni.

- Proxy Server nella backbone

  Questo caso invece, dovrebbe rappresentare il caso bilanciato; infatti, se i due nodi comunicanti fossero entrambi nodi mobili, il tempo impiegato da entrambi per effettuare la fase di update sarebbe circa lo stesso.

- Proxy Server nella rete del MN

  Questo è il caso in cui, il tempo impiegato dal nodo mobile per aggiornare il server proxy ABPS dei suoi spostamenti è il minore. Si suppone che il nodo mobile si muova in reti periferiche limitrofe a quella dove si trova il proxy server ABPS. Si prevede quindi che questo sia il caso in cui gli intervalli di indisponibilità siano minori rispetto alle altre configurazioni.

5.2.1 Scenario Test

Lo scenario test presenta una topologia simmetrica, dove sono presenti quattro access point disposti ai quattro vertici di un quadrato. La configurazione prevede che il nodo mobile si muova tra le reti di questi access point in moto circolare uniforme, con centro della circonferenza corrispondente al centro del quadrato formato dai quattro access point. In questo scenario si ha la possibilità di testare la prestazione del protocollo sotto analisi, in un'ambiente spoglio, che mostri i risultati ideali. La Figura 5 mostra la struttura della rete Test come è stata descritta precedentemente: la circonferenza azzurra è il tragitto percorso dal nodo mobile durante la simulazione; mentre il quadrato rosso è la struttura regolare, ai quali vertici sono stati posizionati gli access point.

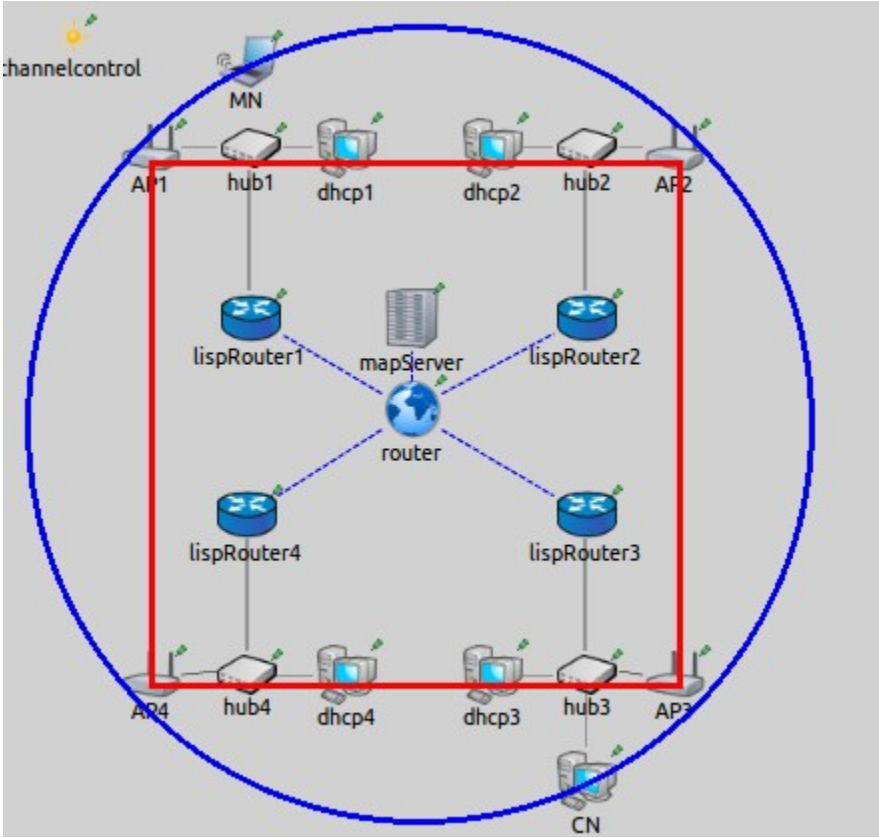


Figura 5 - Esempio di rete utilizzata per i test.

La tabella 3 mostra l'immagine, e descrive le principali caratteristiche, delle due reti configurate per l'architettura LISP. Nelle immagini all'interno della tabella, sono specificati di colore rosso i prefissi EID per i quali i router LISP sono autoritativi. Si nota come l'implementazione del simulatore LISP abbia limitato l'impostazione della rete alla sola presenza di un unico Map Server collegato alla backbone. In un caso più verosimile, ogni sito LISP (o LISP router) sarebbe dovuto essere in grado di configurare il proprio Map Server; questo si suppone nei pressi del sito a cui fa riferimento; il suo collocamento nella rete di interconnessione introduce invece un'ulteriore latenza (di un RTT), anche per le Map Request relative a prefissi EID o EID per i quali il sito LISP dovrebbe essere autoritativo.

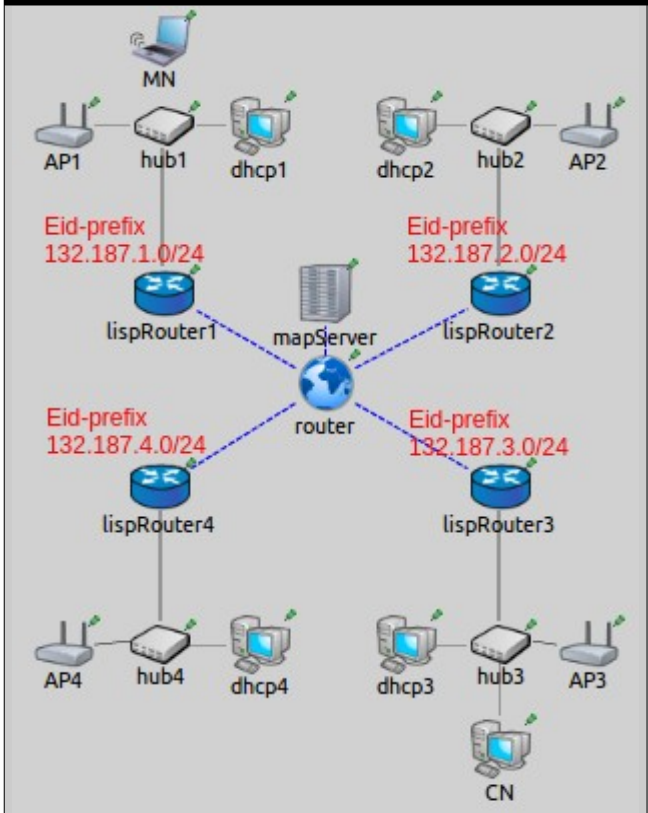
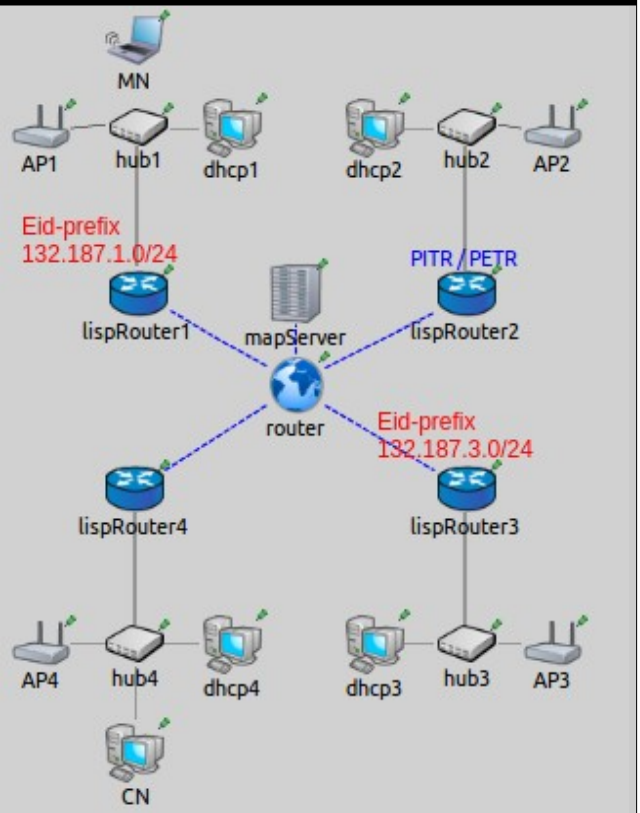
CN in rete LISP	CN in rete non-LISP
	
Il nodo destinatario CN si trova sotto il sito LISP per cui *lispRouter3* ricopre la funzione di ITR e ETR.	Il nodo destinatario CN si trova sotto un sito non-LISP che per inviare e/o ricevere LISP necessità del servizio del Proxy ITR/ETR *lispRouter2*.

Tabella 3

La tabella 4 invece mostra le due configurazioni sperimentate per l'architettura ABPS in questo scenario; la terza specificata in precedenza non è stata applicata in quanto, il nodo mobile, entrando o uscendo da una delle reti periferiche o è a distanza di un RT (Round Trip) attraverso la backbone o si trova nella stessa rete; dunque non è possibile impostare lo scenario in modo che si verifichi la vicinanza con il MN per ogni suo spostamento.

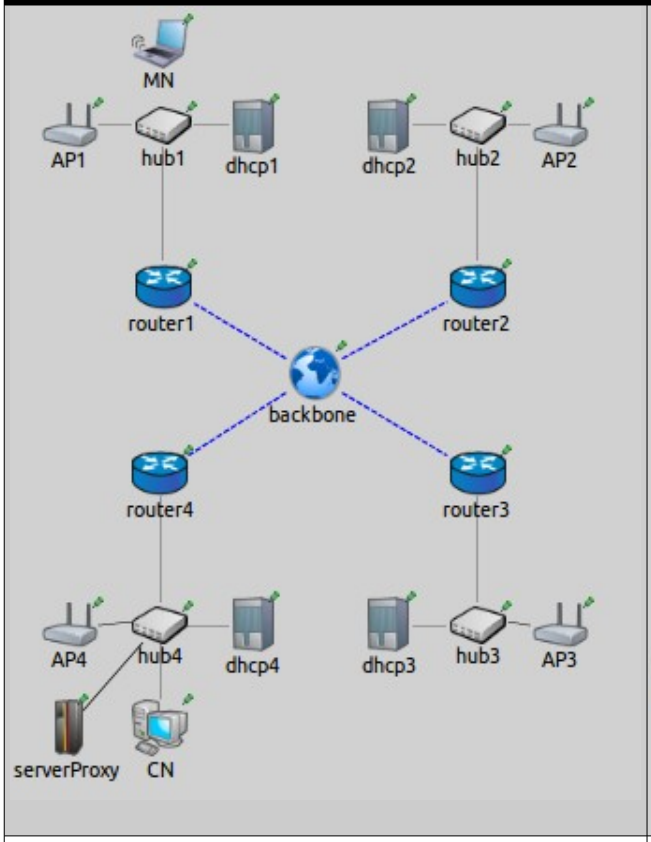
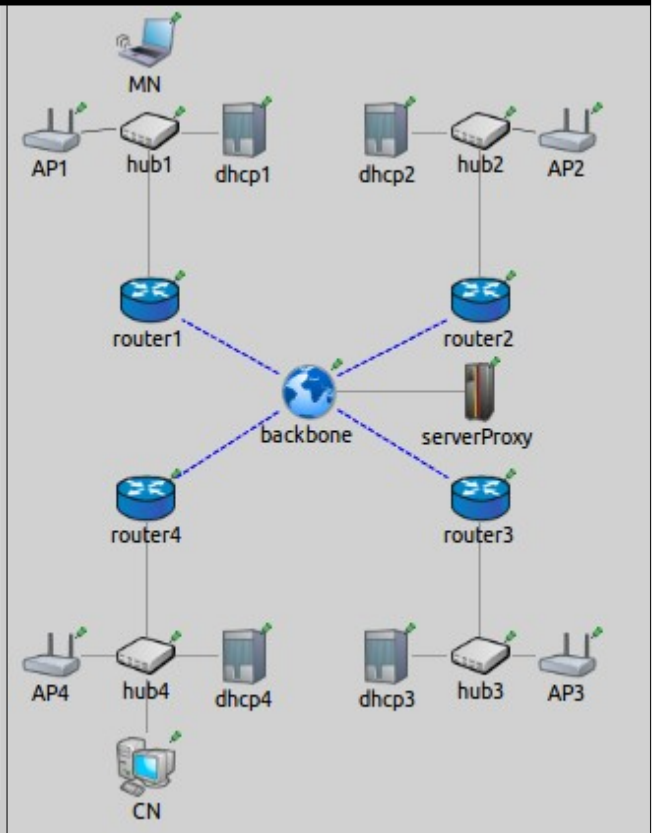
Proxy Server vicino al CN	Proxy Server nella Backbone
	
Il Proxy Server ABPS *serverProxy* si trova nella rete periferica nella quale è collocato anche il *CN*.	Il Proxy Server ABPS *serverProxy* si trova nella rete di interconnessione, a uguale distanza da *CN* e *MN*.

Tabella 4

5.2.2 Scenario Urbano

Lo scenario urbano permette, rispetto a quello test, di valutare un ambiente dove il nodo mobile e il CN si trovano in due reti di classe B differenti (128.96.0.0/16 e 128.97.0.0/16); queste due reti si suppone siano distanti tra di loro, ma sotto la stessa rete di classe A. I due network sono collegati anche attraverso la rete di interconnessione (backbone) ad alta latenza, che qui rappresenta un qualsiasi instradamento passante per la backbone che partendo dalla rete del MN, arriva alla rete del CN (128.97.0.0/16). La rete è stata strutturata in modo che il MN possa effettuare roaming tra le reti appartenenti a 128.96.0.0/16 senza che queste siano separate da un collegamento ad alta

latenza, come accadeva per lo scenario di prova. Si tratta dunque di una rete periferica che comprende diverse sotto reti connesse tra loro da link a bassa latenza (ordine dei microsecondi). La latenza dei collegamenti azzurri è la metà (30ms) di quella dei collegamenti grigi (60ms), per permettere la percorrenza da una rete di classe B all'altra nello stesso tempo indipendentemente dal percorso scelto.

La figura 6 mostra come sia gerarchicamente strutturato lo scenario, dal punto di vista delle reti che lo compongono; i diversi riquadri rappresentano ognuno una rete diversa; quelli rossi definiscono le reti di classe B; quelli marroni definiscono le reti di classe C e quelli oro definiscono le reti subnettate a 25 bit (maschera di rete 255.255.255.128). Questa è dunque la struttura della rete sulla quale si sono configurati i due simulatori.

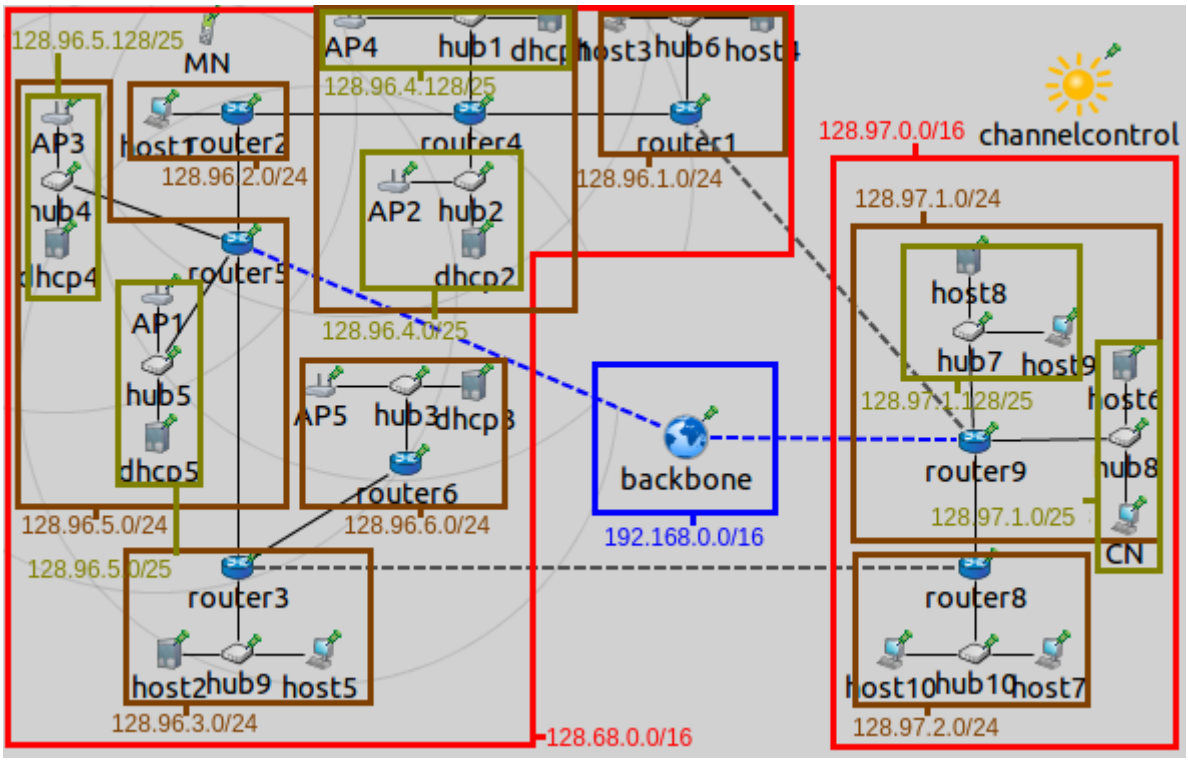


Figura 6 - Struttura della rete urbana utilizzata per il confronto.

A questra struttura è possibile aggiungere alla simulazione un livello aggiuntivo di ostacoli; questi sono disposti come lo sarebbero i muri di alcune abitazioni. Nella Figura 7, i riquadri neri rappresentano la disposizione degli ostacoli e, il percorso definito dalla linea viola è il tragitto che il nodo mobile compie durante la simulazione nello scenario urbano. Durante il percorso, che siano presenti gli ostacoli o meno, il MN effettuerà diversi handover orizzontali tra gli access point sul tragitto, che rimangono quelli visibili in Figura 6.

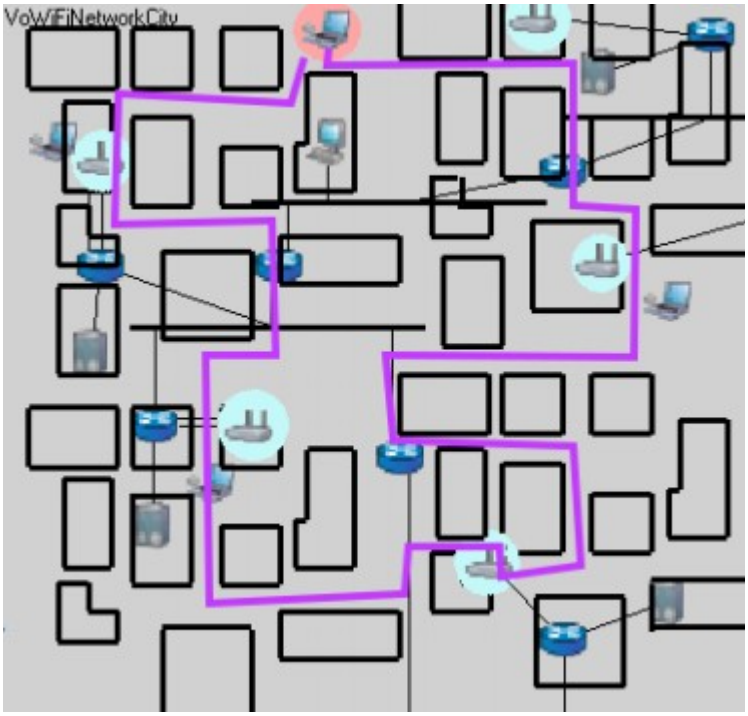


Figura 7 - Disposizione degli ostacoli e percorso del MN durante la simulazione.

Definito lo scenario trattato nel confronto urbano, si può descrivere come le configurazioni per i vari protocolli di supporto alla mobilità sono state integrate con lo stesso, per poter essere valutate. Per LISP sono stati impostati i due scenari mostrati in Figura 8 ed in Figura 9; queste mostrano chiaramente come il *deployment* dell'architettura LISP sia una sovrastruttura rispetto alla rete sottostante, che non deve essere modificata per ospitare le componenti LISP. L'unica parte che risulta onerosa per la configurazione dello scenario, che

utilizza il PITR/PETR, è la modifica delle varie routing table dei router per instradare i pacchetti diretti a siti LISP (altrimenti non routabili per definizione) verso il PITR/PETR, che interfacciandosi con l'infrastruttura LISP potrà portare a termine la comunicazione con il destinatario.

Per la prima configurazione, il correspondent node si trova sotto ad una rete LISP, ed il suo indirizzo IP vale come se fosse un EID, con prefisso 128.97.1.0/24; infatti il dominio degli EID è stato impostato a 128.0.0.0/8, permettendo anche alla rete di calsse B, sotto la quale risiede il CN di farne parte. Nella seconda configurazione invece, il dominio degli EID si restringe a 128.96.0.0/16, che rappresenta tutto l'insieme degli indirizzi della rete di classe B sotto la quale risiede il nodo mobile. Il CN invece possiede un indirizzo non routabile LISP, in quanto non fa parte del dominio EID configurato. La sua esclusione dall'infrastruttura LISP lo costringe a fare affidamento su un Proxy ITR/ETR per l'inoltro e la ricezione a/da siti LISP; questo ruolo è ricoperto in Figura 9 da lispRouter4, che funge da PITR/PETR per gli indirizzi non-LISP dello scenario.

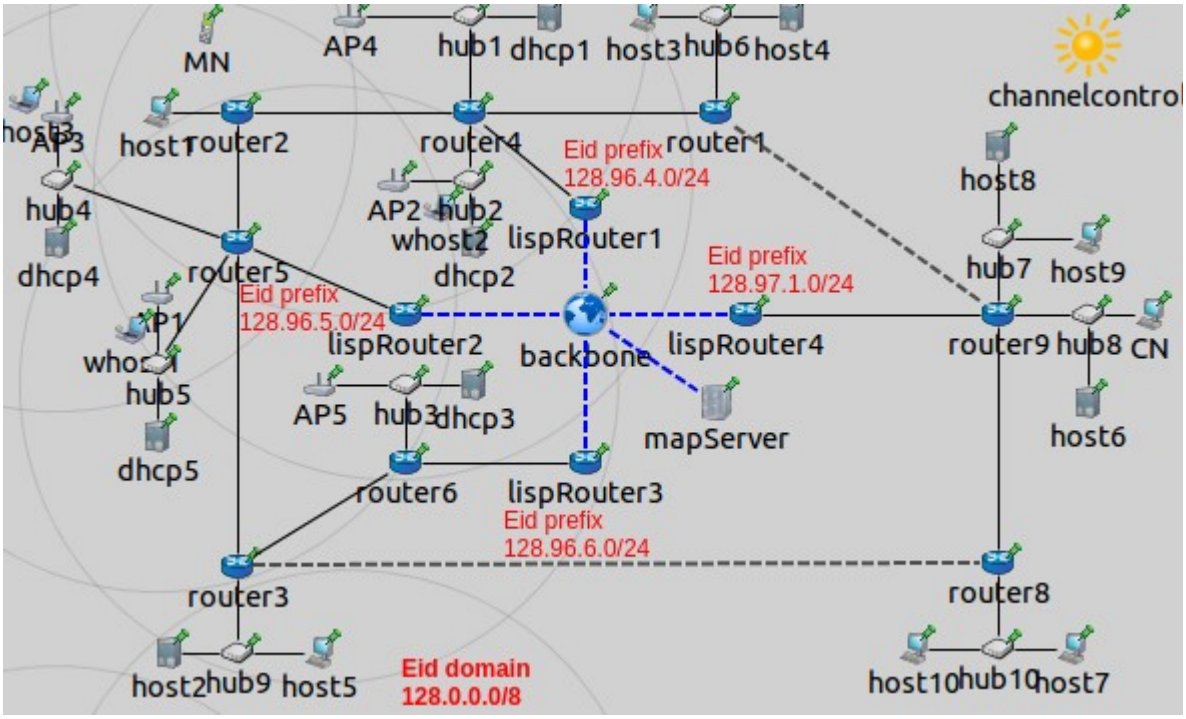


Figura 8- Configurazione con il CN in un sito LISP (a destra).

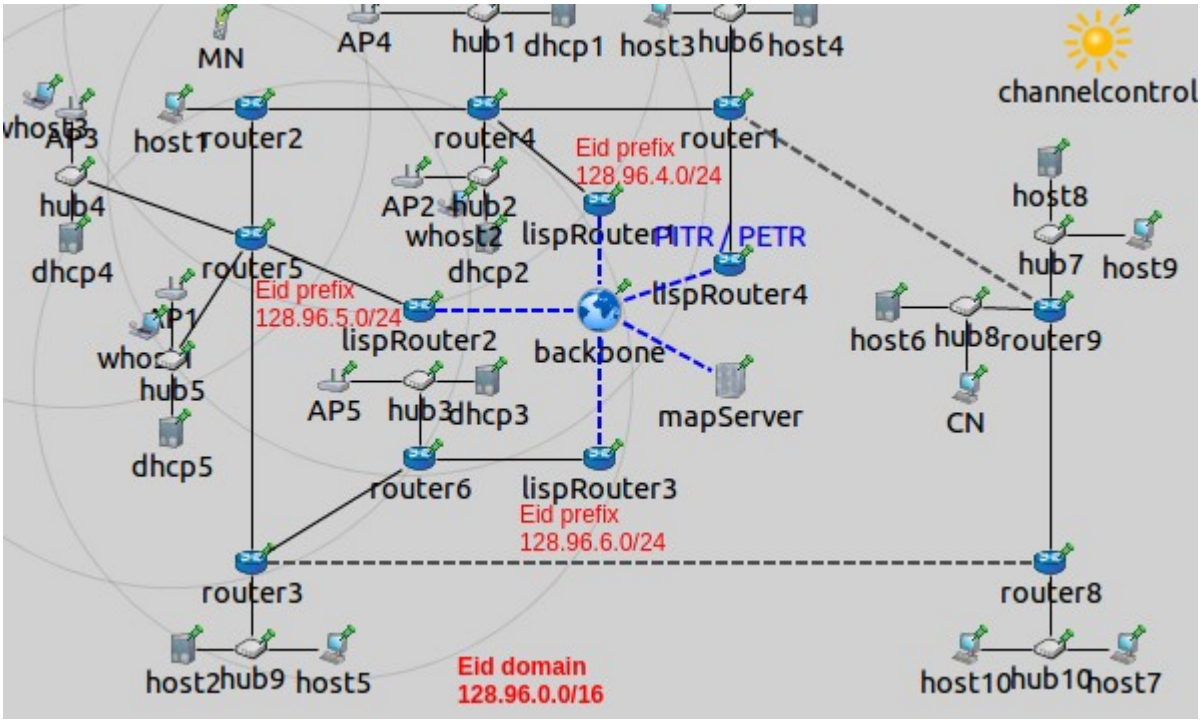


Figura 9 - Configurazione con il CN in un sito non-LISP.

Nelle tre figure 10, 11 e 12, vengono mostrate le reti dello scenario urbano per le diverse configurazione del protcollo ABPS. A differenza di LISP, le componenti inserite sono inferiori, in quanto il protocollo stesso comporta dei cambiamenti all'infrastruttura già esistene molto ridotti rispetto a LISP. Non sono aggiunti router o ridefinite le semantiche degli indirizzi di rete, ma è stato aggiunto un solo proxy server ABPS nelle diverse posizioni discusse in 5.2, che è sufficiente per lo scopo della simulazione. Il traffico è sempre passato tra le due reti periferiche (Figura 11 e 12), senza dover percorrere la rete di interconnessione; che è presente solamente nel caso in cui il proxy server vi risieda (Figura 10). Questo perchè a differenza di LISP, ABPS non presenta un database di mapping distribuito (Map Server). Prendendo in considerazione un RTT, il tempo necessario per percorrere la backbone, andata e ritorno, il tempo impiegato dal nodo mobile per comunicare al proxy server il suo nuovo indirizzo IP nelle diverse configurazioni variano seguendo questi schemi:

- Figura 10 - 1 * RTT / 2 = ~0.06s
- Figura 11 - 1 * RTT = ~0.12s
- Figura 12 - qualche microsecondo.

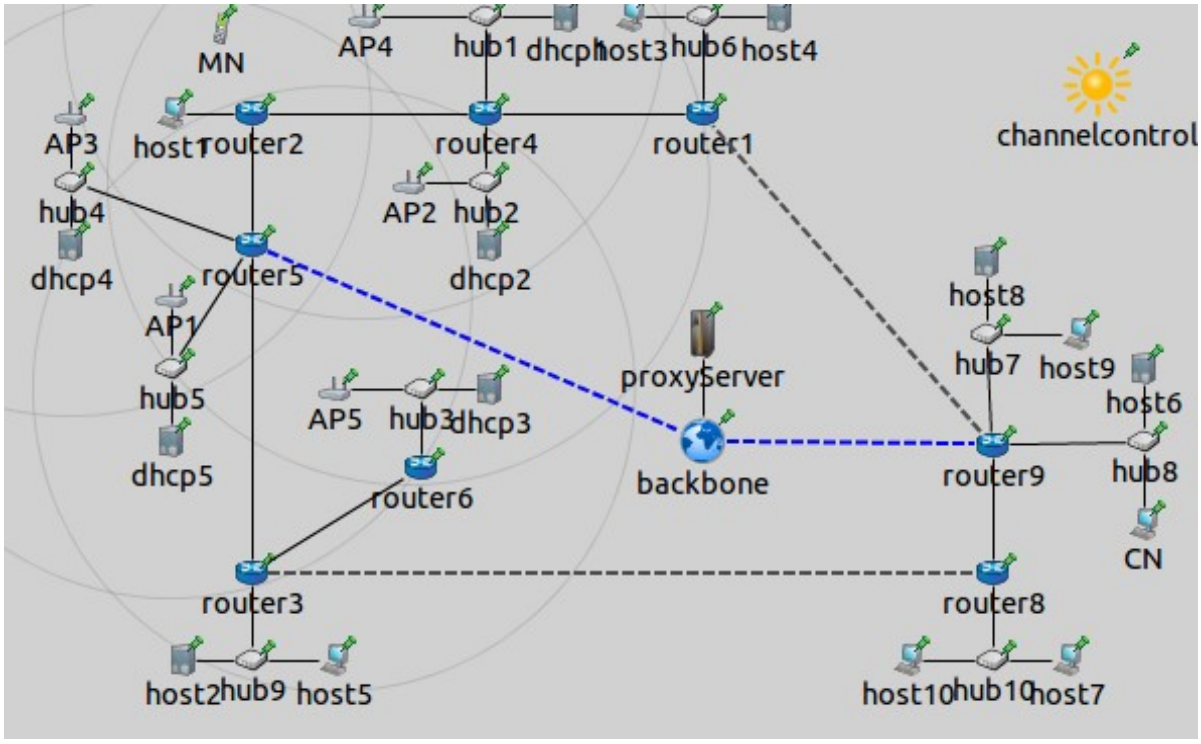


Figura 10 - Configurazione con il Proxy Server ABPS nella backbone.

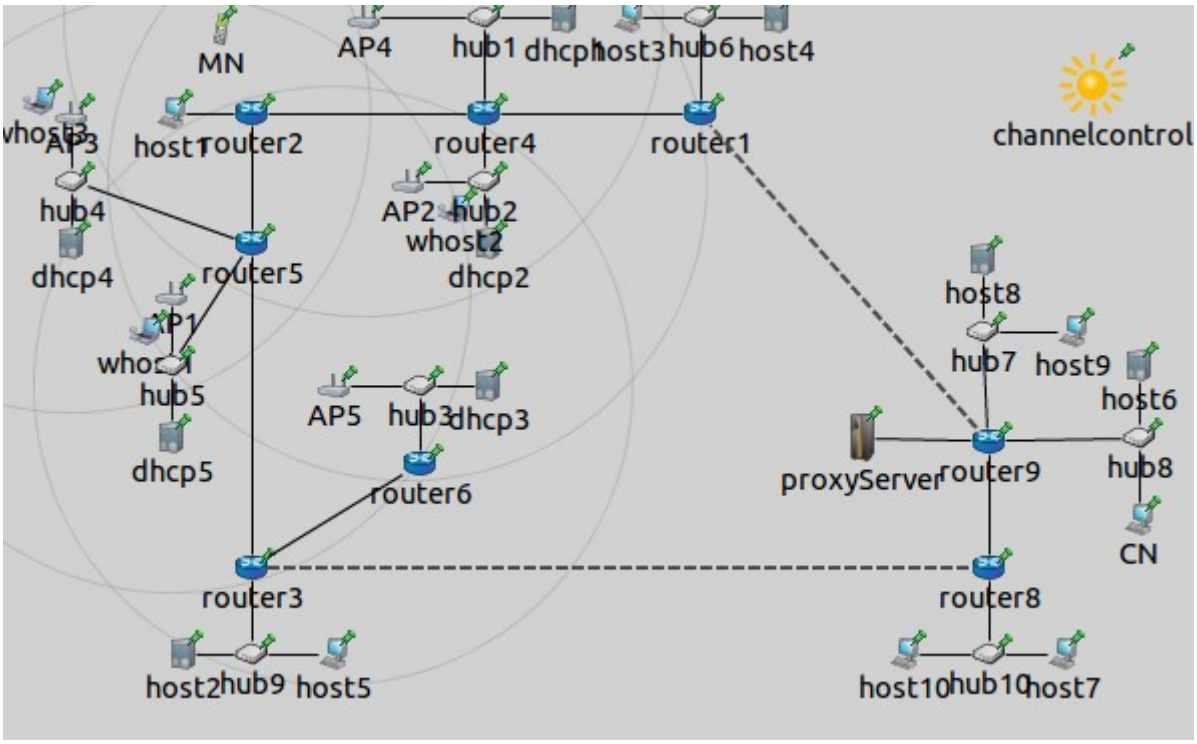


Figura 11 - Configurazione con il Proxy Server ABPS nella rete del CN.

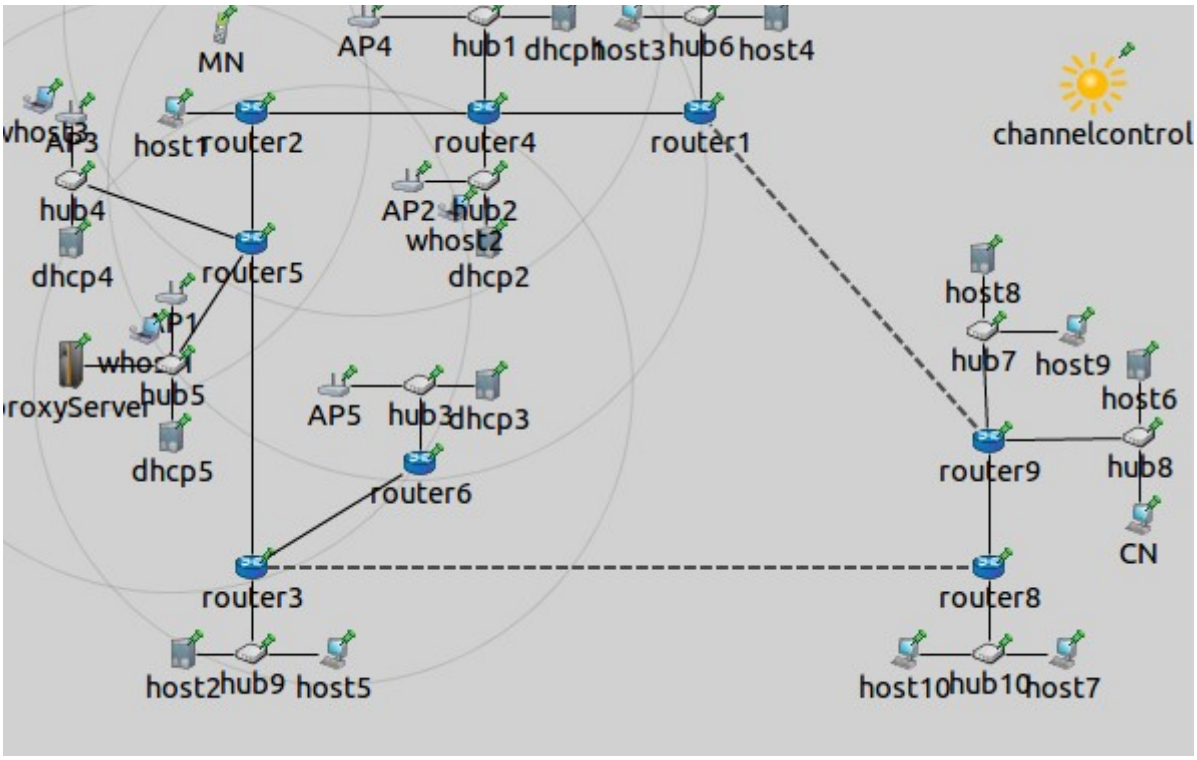


Figura 12 - Configurazione con il Proxy Server ABPS nella rete (di classe B) del MN.

5.3 Parametri e configurazioni

Per ognuno degli scenari e per ognuno dei protocolli sono stati impostati diversi parametri e configurazioni. I parametri per le simulazioni sono impostati tramite file *ini*, che si trova nella directory principale di ogni scenario. Quelli comuni ad entrambi gli scenari sono elencati di seguito:

```
# statistical recording parameters
**.debug = true
**.udpApp*.packet delays.vector-recording = true
**.udpApp*.sender handover downtimes.vector-recording = true
**.udpApp*.receiver handover downtimes.vector-recording = true
**.vector-recording = false

# channel physical parameters
*.channelcontrol.probUnidirectional = 0

# access point
**.mgmt.frameCapacity = 10
```

```
# wireless configuration
**.wlan*.agent.activeScan = true
**.wlan*.agent.probeDelay = 0.1s
**.wlan*.agent.minChannelTime = 0.15s
**.wlan*.agent.maxChannelTime = 0.3s
**.wlan*.agent.authenticationTimeout = 5s
**.wlan*.agent.associationTimeout = 5s

# mac configuration
**.mac.address = "auto"
**.mac.maxQueueSize = 14
**.wlan.mac.retryLimit = 7
**.wlan.mac.cwMinData = 7
**.wlan.mac.cwMinBroadcast = 31

# radio configuration
**.radio.thermalNoise = -110dBm
**.radio.pathLossAlpha = 2
**.radio.snirThreshold = 4dB
```

Il primo blocco di parametri definisce i vettori di dati che si vogliono registrare durante la simulazione; registrare tutti i vettori, anche quelli che non servono nella propria analisi, per test onerosi può comportare file di risultati molto pesanti. Successivamente vengono definiti altri parametri per il canale fisico, il livello mac e il segnale radio. Altri settaggi da tenere in considerazione sono quelli che riguardano il multihoming e le applicazioni, dunque, numero interfacce, frequenza di invio e grandezza dei messaggi che per entrambi i simulatori, per coerenza, sono stati impostati equivalenti:

- *wlans* - 1, 2
- *messageFreq* - 20ms, 40ms
- *messageLength* - 512 byte

Per i vari simulatori sono state create, in modo che possano essere selezionate a piacimento al momento del lancio dei test, le seguenti configurazioni nel file *omnetpp.ini*:

- LISP
  - CN-LISP - il correspondent node (CN) si trova in una rete LISP.

- CN-PITR - il correspondent node (CN) si trova in una rete non-LISP, viene utilizzato un Proxy ITR/ETR per la comunicazione da/a siti LISP.

- ABPS
  - EXT-PS - il Proxy Server (PS) è collocato lungo la backbone.
  - CN-NEAR-PS - il Proxy Server (PS) si trova nella stessa rete del CN.
  - PS-NEAR-MN - il Proxy Server (PS) si trova nella stessa rete del MN.

5.3.1 Scenario Test

Per lo scenario test, oltre ad avere una rete con architettura LISP ed una rete con architettura ABPS, è stato aggiunto il caso in cui il nodo mobile LISP (Lisp-MN) sia dotato di più interfacce, quindi multihomed, e le utilizzi concorrentemente per inviare i pacchetti; questo meccanismo è del tutto simile al comportamento del proxy client ABPS, ed è un buon punto di confronto per valutare il supporto per la comincazione multihoming, per il nodo mobile che effettua roaming, delle due architetture. Le configurazioni impostate per lo scenario test di LISP in gerale sono i seguenti:

```
[Config CN-PITR]
network = CN_PITR
description = "CN is deployed in a non-LISP site."
**.mapResolver.PITRModule = "lispRouter2"
**.lisp.PETRAddress = "192.168.0.2"
**.lisp.eidPrefix = "132.187.2.0"
**.lisp.eidPrefixLength = 22

[Config CN-LISP]
network = CN_LISP
description = "CN is deployed in a LISP site."
**.usePETR = false
```

I parametri impostati per il simulatore di ABPS, invece, descrivono in modo analogo le configurazioni descritte precedentemente per questo protocollo. Nell'elenco delle configurazioni impostate sono state omesse quelle che permettono di variare la frequenza con cui vengono spediti i messaggi dalle

applicazioni, in quanto sarebbero state ridondanti. Per lo scenario test di ABPS sono inoltre configurati due casi aggiuntivi, dove uno dei due nodi comunica con una frequenza molto maggiore dell'altro; questo per permettere di valutare se questa caratteristica può incidere sulla capacità di update di ABPS e dunque sulla lunghezza dei suoi intervalli di indisponibilità:

```
[Config CN-NEAR-PS]
network = CN_NEAR_PS
description = "Proxy Server is deployed near CN."
**.MN.udpApp[0].proxyServerAddress = "132.187.4.7"
**.CN.udpApp[0].proxyServerAddress = "132.187.4.7"

# il nodo mobile manda i messaggi con frequanza molto minore del CN
[Config MN_LOW_MSG_FREQ]
extends = CN-NEAR-PS
description = "MN sends messages interval is >> than CN."
**.MN.udpApp[0].messageFreq = 1s

# il CN manda i messaggi con frequanza molto minore del nodo mobile
[Config CN_LOW_MSG_FREQ]
extends = CN-NEAR-PS
description = "CN sends messages interval is >> than MN."
**.CN.udpApp[0].messageFreq = 1s

[Config EXT-PS]
network = EXT_PS
description = "Proxy Server is deployed in the backbone."
**.MN.udpApp[0].proxyServerAddress = "192.168.0.6"
**.CN.udpApp[0].proxyServerAddress = "192.168.0.6"
```

Per quanto riguarda la mobilità del terminale, è stata impostata in modo che il nodo mobile segua un andamento circolare uniforme attorno al centro dello scenario; il MN inizia il proprio movimento partendo in corrispondenza del primo spigolo (in alto a sinistra) del quadrato definito dagli access point:

```
# mobility
**.MN.mobilityType = "CircleMobility"
**.MN.mobility.speed = 2 mps
**.MN.mobility.startAngle = -135deg
**.MN.mobility.r = 175
**.MN.mobility.cx = 275
**.MN.mobility.cy = 275
```

5.3.2 Scenario urbano

Per lo scenario urbano la maggior parte delle impostazioni sono rimaste invariate, se non qualche cambio di indirizzo alle reti e l'inserimento della configurazione in cui il proxy server ABPS è nella stessa rete del nodo mobile:

```
[Config EXT-PS]
network = EXT_PS
description = "Proxy Server is deployed in the backbone."
**.MN.udpApp[0].proxyServerAddress = "192.168.1.3"
**.CN.udpApp[0].proxyServerAddress = "192.168.1.3"

[Config EXT-PS_no-obstacles]
extends = EXT-PS_20ms
**.ob*.penetrationLoss = -0dBm
```

La configurazione sopra riportata mostra una nuova caratteristica di questo scenario, infatti è possibile impostare la *penetrationLoss* in modo da rendere gli ostacoli nulli, senza doverli onerosamente rimuovere a mano ogni volta. Il valore di default di *penetrationLoss* degli ostacoli è -2dBm; questa possibilità è stata impostata per ogni configurazione di LISP e ABPS. Viene modificata la mobilità del terminale, importando un file xml che descrive il percorso mostrato in Figura 7, che prevede uno spostamento costante di 1mps:

```
# mobility
**.MN.mobilityType = "TurtleMobility"
**.MN.mobility.turtleScript = xmldoc("constPath.xml", "pre//nohtml//movement")
```

Sono stati impostati i 14 canali radio, i quali sono scansionati con parametri successivamente mostrati; questi però aumentano la durata dell'intervallo di indisponibilità, per nodi non multihomed, perchè per effettuare una scansione di tutti i canali impiegano nel caso peggiore ~4.2s:

```
**.wlan*.agent.minChannelTime = 0.15s
**.wlan*.agent.maxChannelTime = 0.3s
```

Dunque è suggerito, per future sperimentazioni, di utilizzare le seguenti impostazioni, che dovrebbero ridurre il tempo impiegato per la scansione completa, ed avvicinarsi di più ai valori dei casi reali:

```
**.wlan*.agent.minChannelTime = 0.10s
**.wlan*.agent.maxChannelTime = 0.12s
```

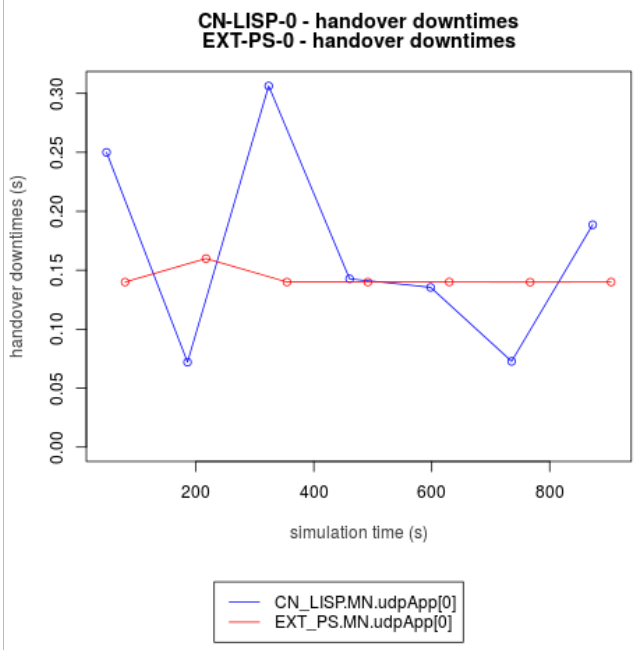
5.4 Risultati

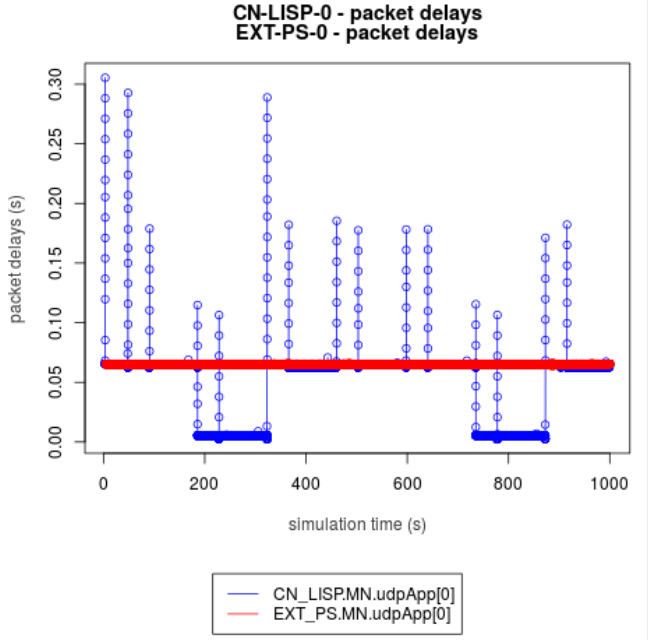
Per analizzare i risultati si sono utilizzati solamente i dati lato mobile node, cioè pacchetti ricevuti, intervallo di indisponibilità in ricezione e latenza dei pacchetti ricevuti da parte del MN. Questo perchè il CN è sempre raggiungibile una volta che il MN si è allacciato ad un access point e ha configurato una delle sue interfacce per la comunicazione; il tempo per fare ciò è lo stesso per tutti i protocolli (variando ovviamente per i casi in cui il nodo mobile è multihomed), in quanto non dipende dall'architettura di supporto alla mobilità utilizzata. I grafici confronteranno le prestazioni della configurazione peggiore per LISP, l'utilizzo di un Proxy ITR e della configurazione peggiore per ABPS, il Proxy Server ABPS nella stessa rete del CN. Mentre la configurazione standard di LISP verrà confrontata con le altre di ABPS. Inoltre viene introdotto il paragone con un terzo protocollo di supporto alla mobilità: MIPv6 [8](Mobile IPv6), che permette al nodo mobile di essere sempre raggiunto tramite il suo Home Address; i pacchetti diretti a questo indirizzo verranno poi inoltrati al nodo mobile in qualunque altra rete si trovi dall'Home Agent, che si trova nella sua home network. L'aspetto di questo protocollo che ne migliorerà le prestazioni in termini di latenza dei pacchetti è la Route Optimization, che permette al nodo mobile e al CN di comunicare direttamente tra loro una volta constatato che sono raggiungibili reciprocamente con la procedura di Return Routability. Di seguito viene riportata una tabella (Tabella 5) dove sono inseriti vari confronti; la prima voce della tabella definisce lo scenario per i quali valgono i grafici

successivi, la seconda definisce le configurazioni confrontate (con le abbreviazioni descritte in 5.3) e la terza definisce alcuni parametri. Prima di ogni grafico vengono specificata la frequenza di invio delle applicazioni, la pesenza o meno di ostacoli nello scenario e il numero (se superiore a uno) delle interfacce del nodo mobile. Non è stata riportata la percentuale dei pacchetti ricevuti con successo in quanto questa, è per la maggior parte un valore che oscilla tra 100% e 99% per tutti i simulatori in situazioni standard, e gli unici casi con molti pacchetti persi sono quelli dove risulta un comportamento anomalo del simulatore. Non è stato possibile inoltre effettuare i test multihomed per LISP nell'ambiente urbano. Le cause sono legate a vari problemi in merito alla scansione dei canali, che in un ambiente complesso come quello urbano, e con più di una sovrapposizione di copertura di access point, sembra non funzionare correttamente. A sinistra sono mostrati gli intervalli di indisponibilità, mentre a destra la latenza dei pacchetti.

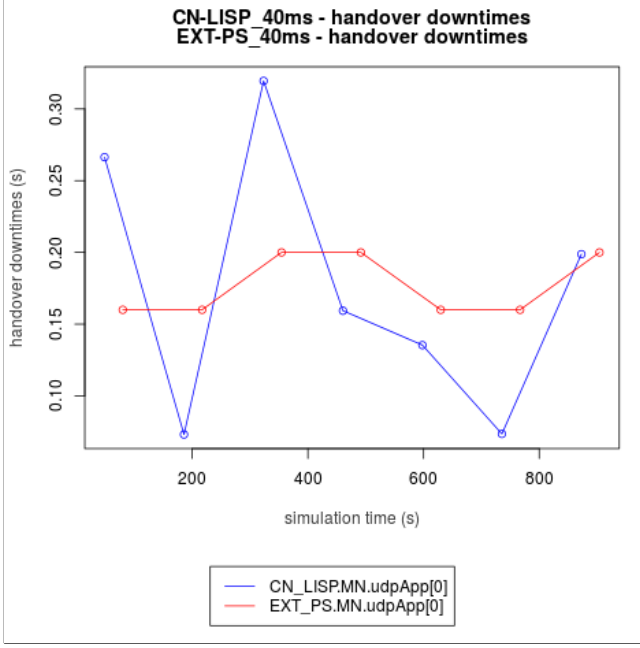
Scenario TEST
CN-LISP vs. EXT-PS

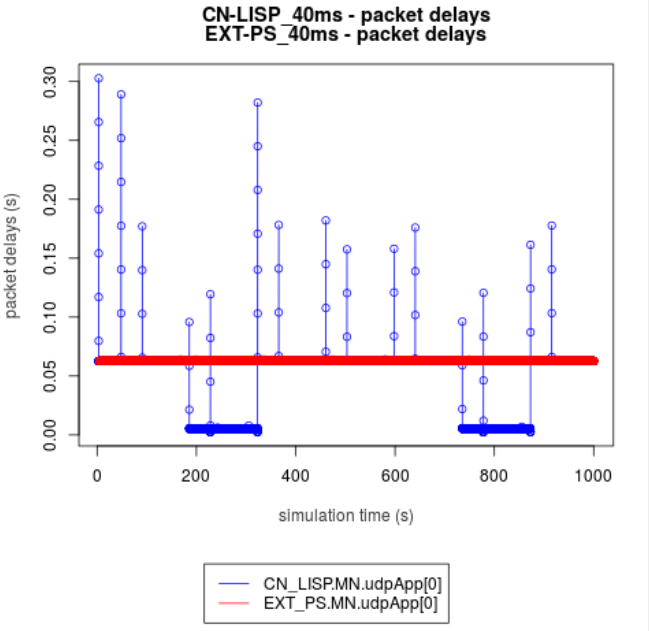
wlans = 2, *messageFreq* = 20ms





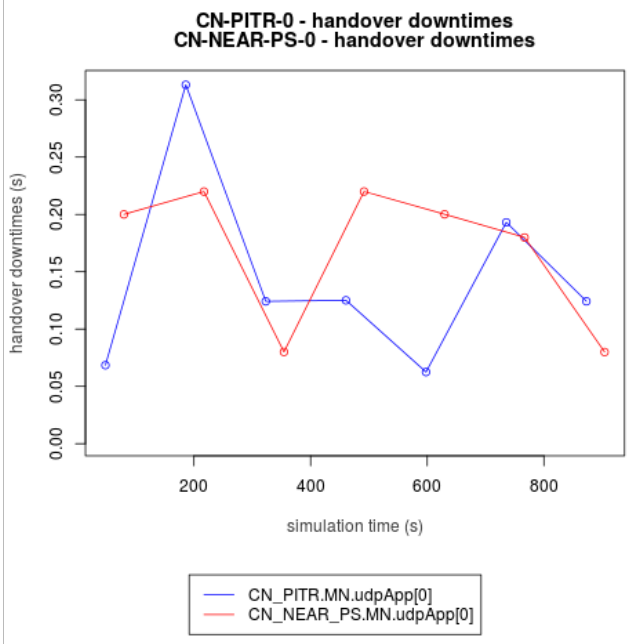
wlans = 2, *messageFreq* = 40ms

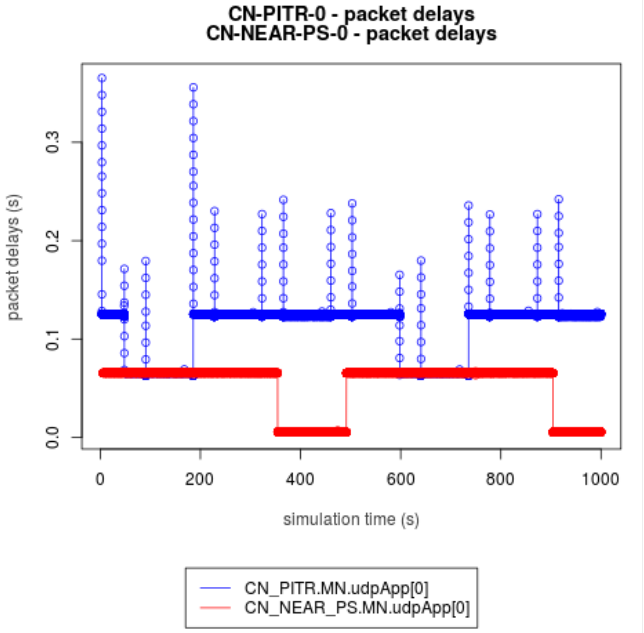




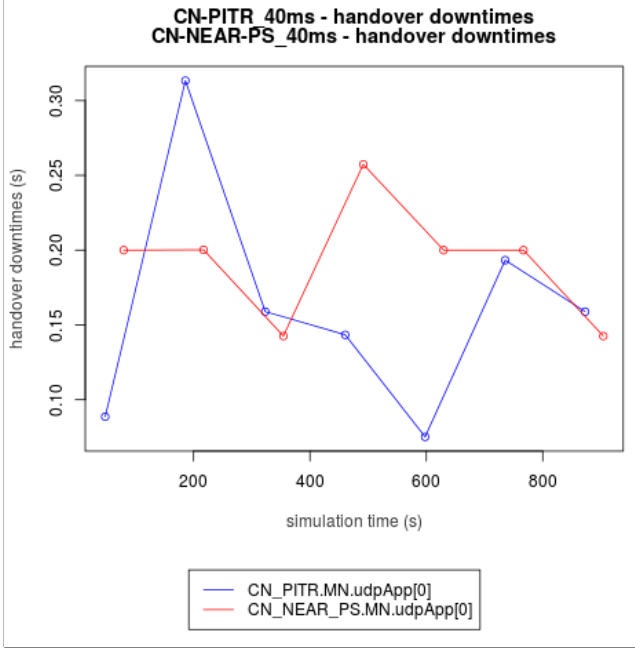
CN-PITR vs. CN-NEAR-PS

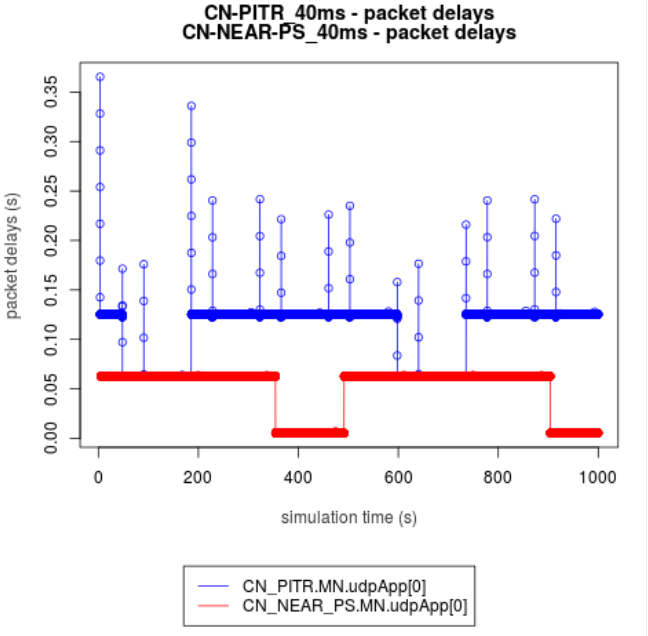
wlans = 2, *messageFreq* = 20ms





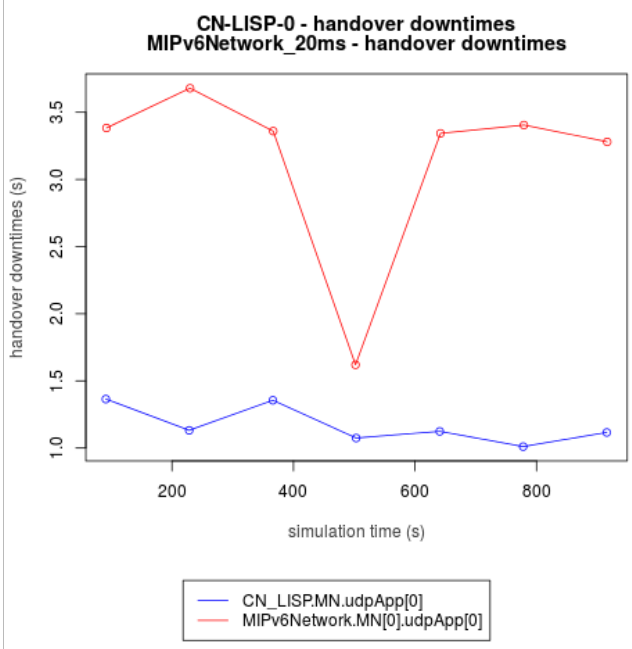
wlans = 2, *messageFreq* = 40ms

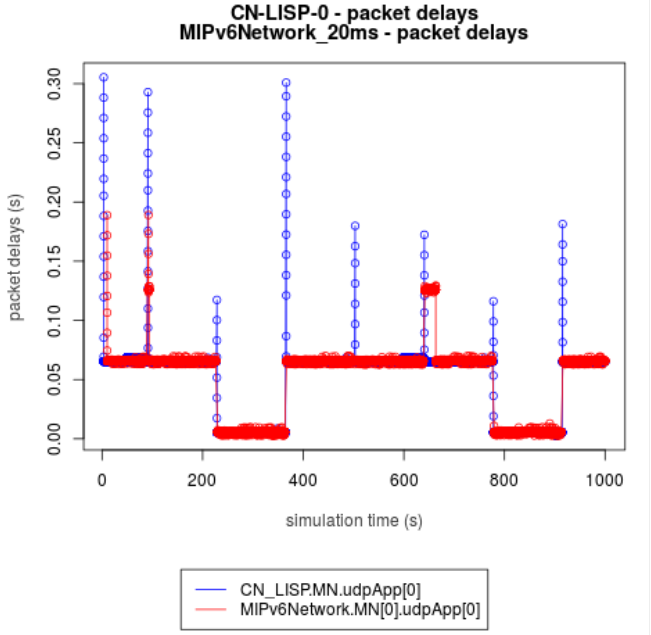




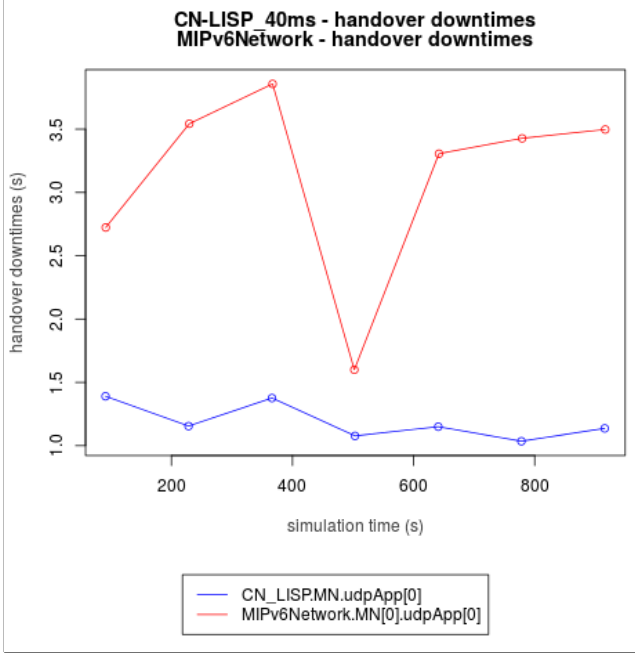
CN-LISP vs. MIPv6

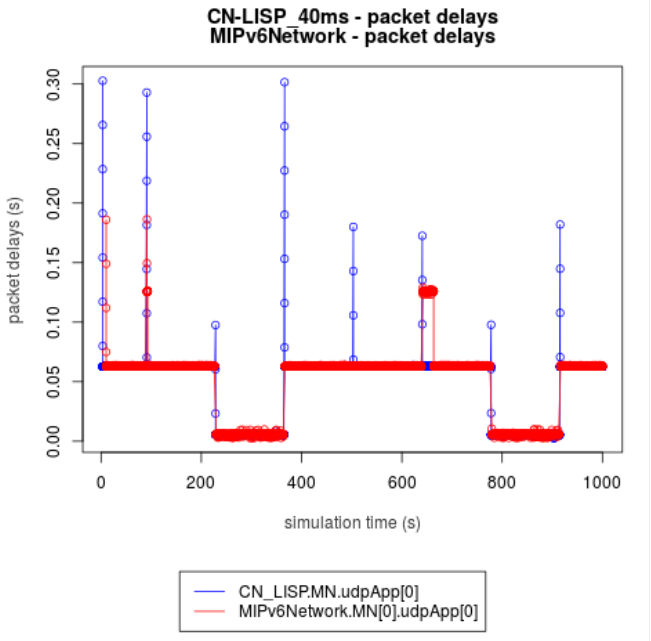
messageFreq = 20ms





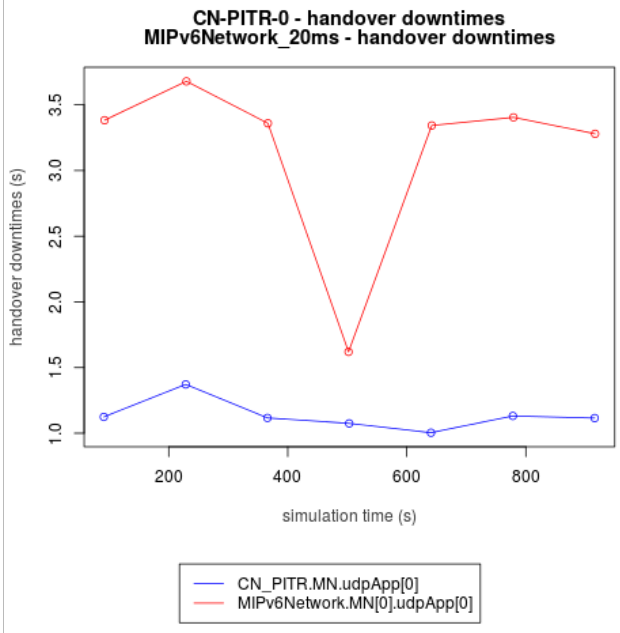
messageFreq = 40ms

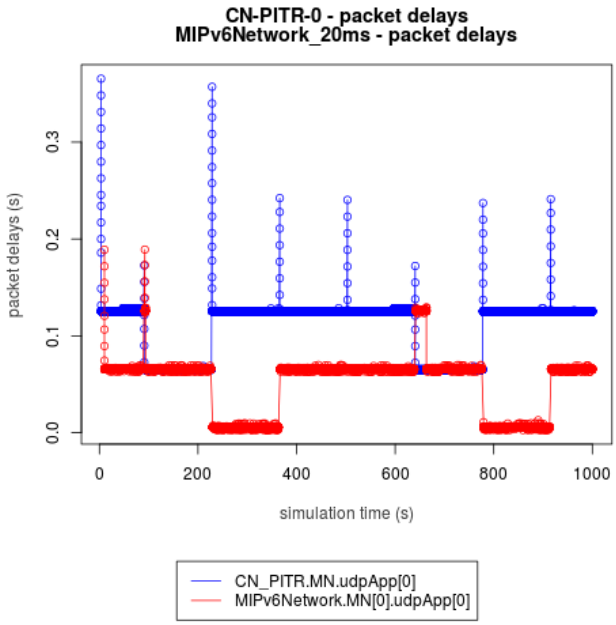




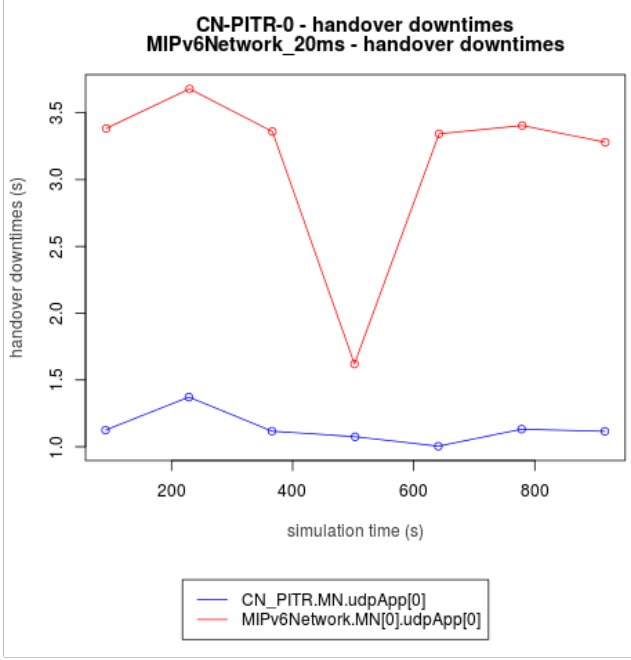
CN-PITR vs. MIPv6

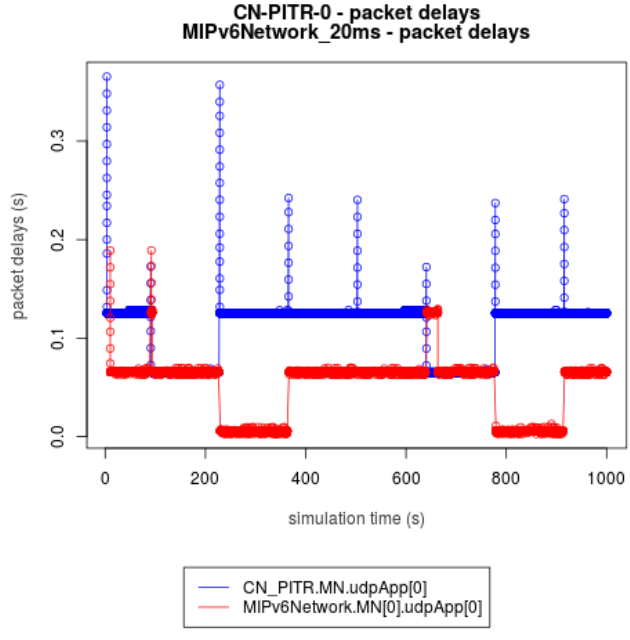
messageFreq = 20ms





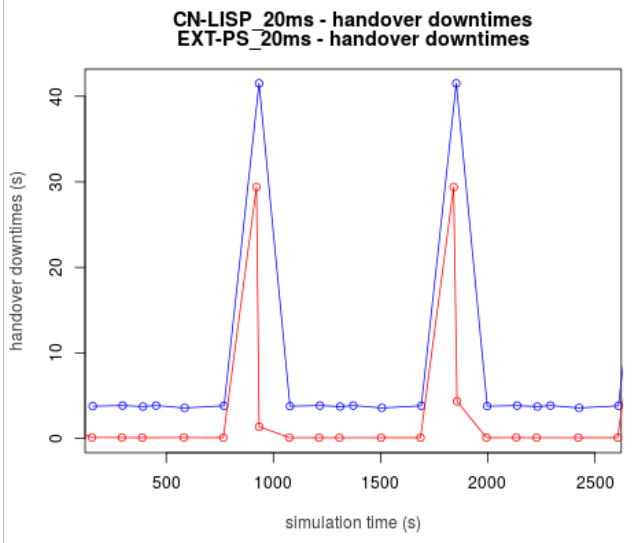
messageFreq = 40ms

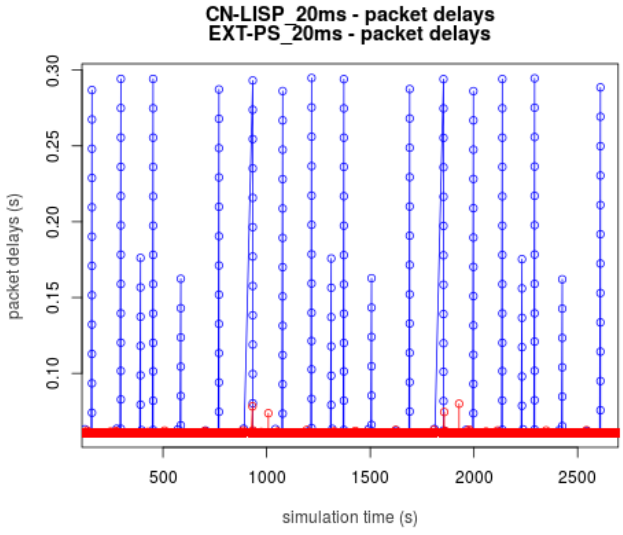




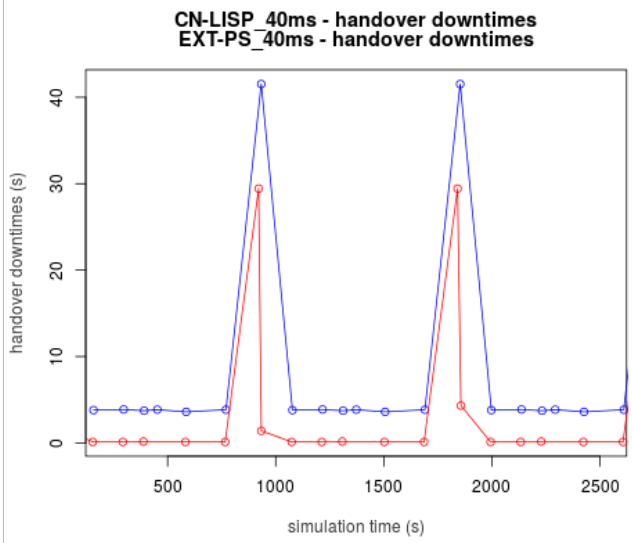
Scenario URBANO
CN-LISP vs. EXT-PS

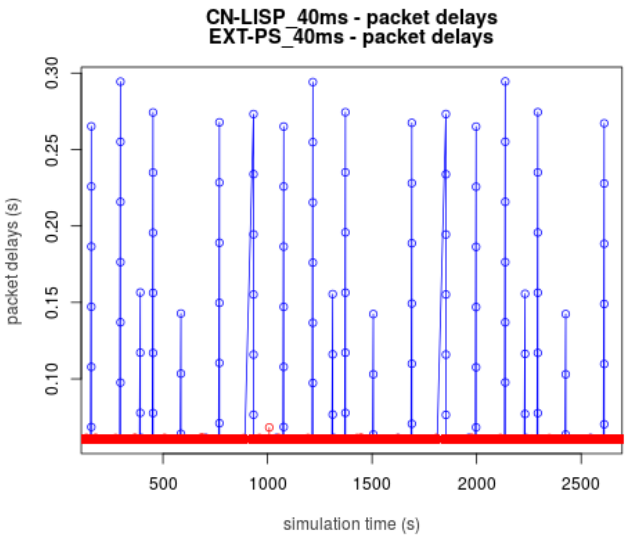
messageFreq = 20ms, *obstacles* = *yes*



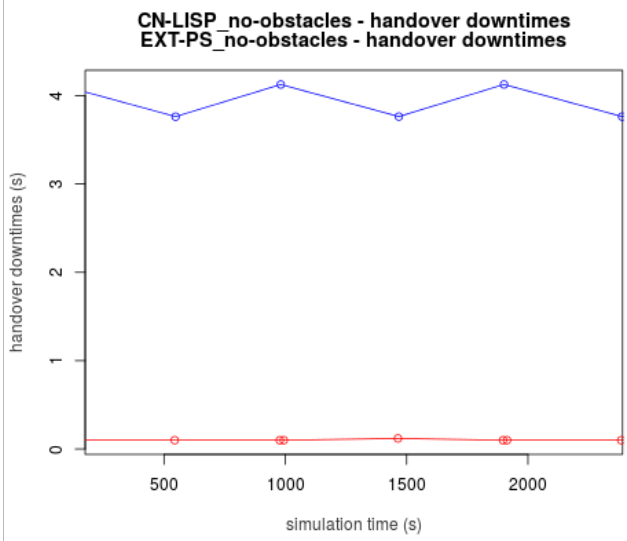


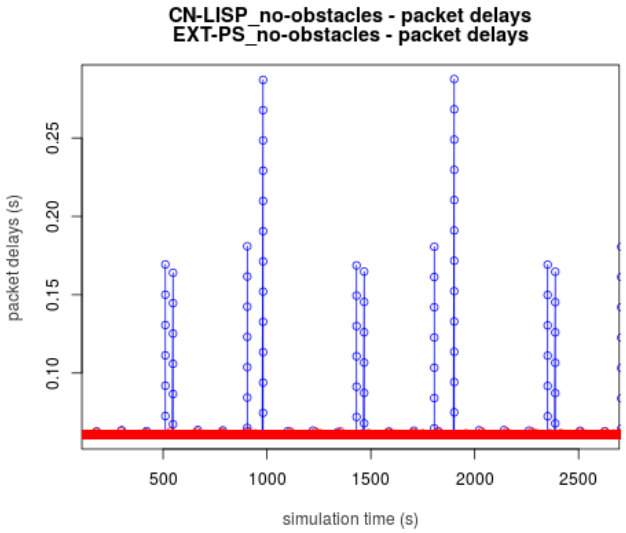
messageFreq = 40ms, *obstacles* = *yes*





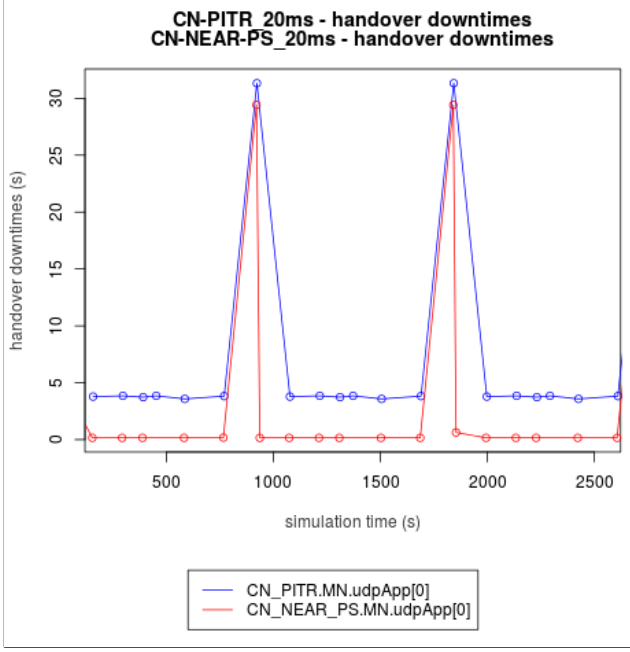
messageFreq = 20ms, *obstacles* = *no*

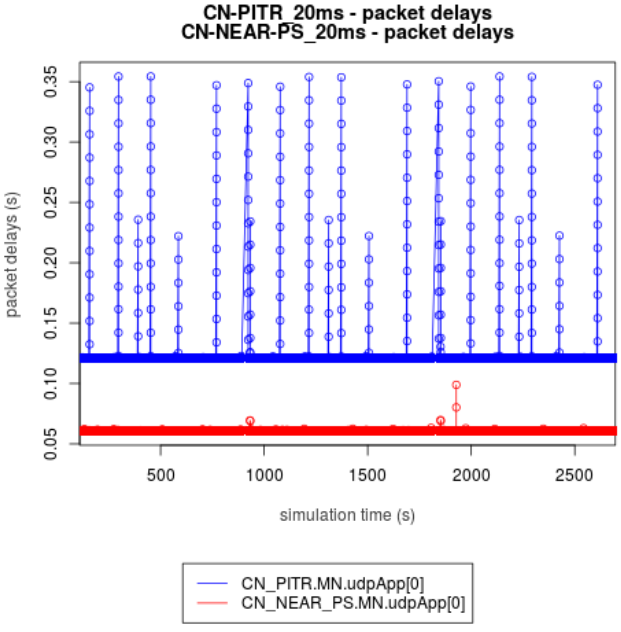




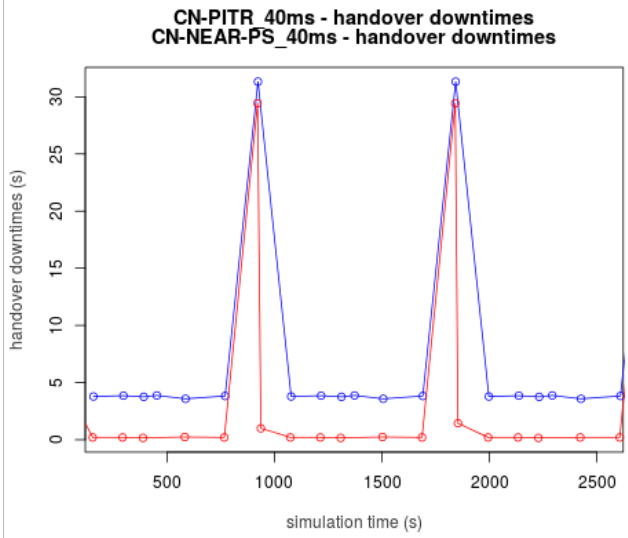
CN-PITR vs. CN-NEAR-PS

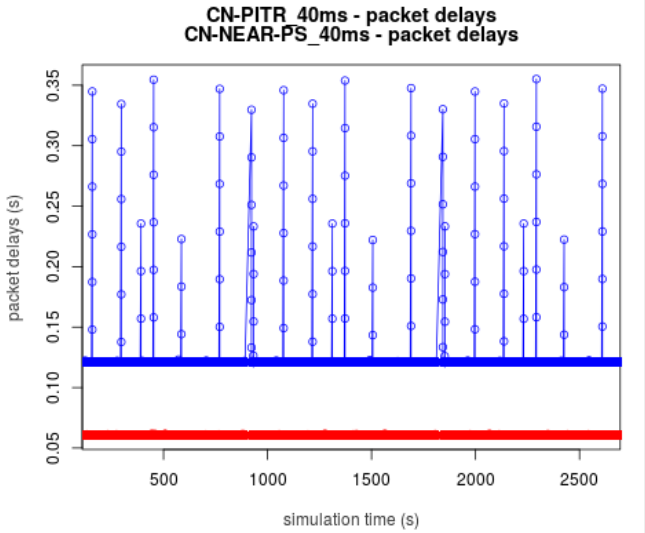
messageFreq = 20ms, *obstacles* = *yes*



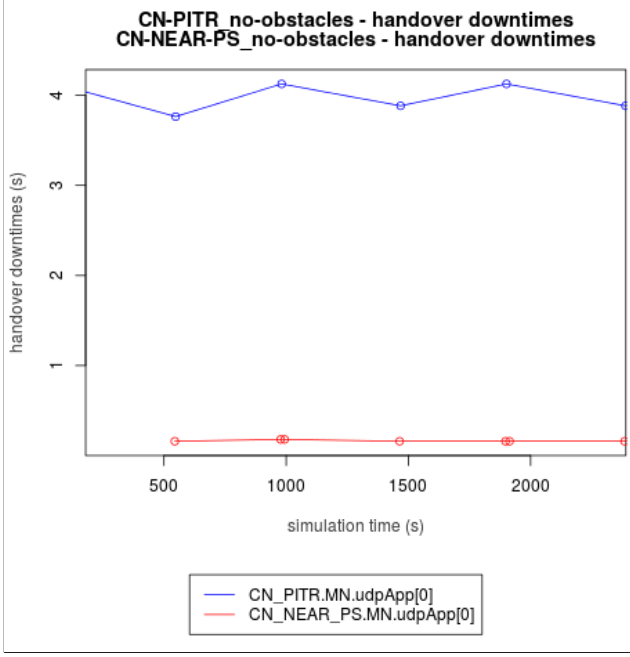


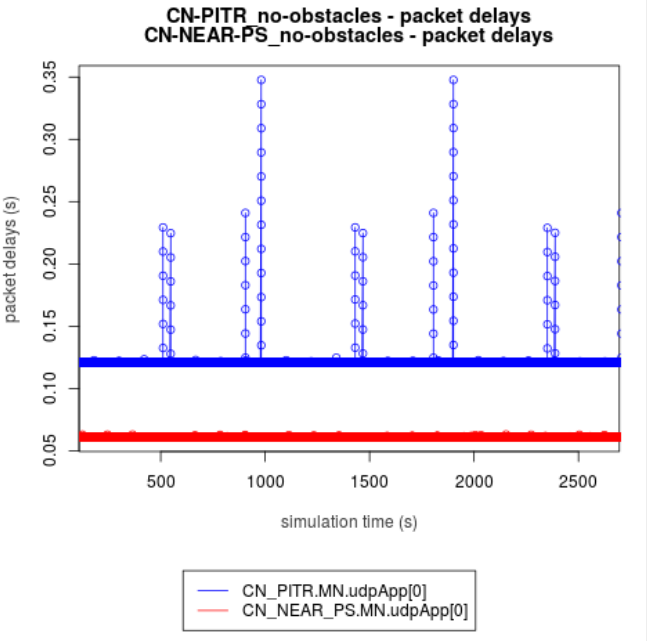
messageFreq = 40ms, *obstacles* = *yes*





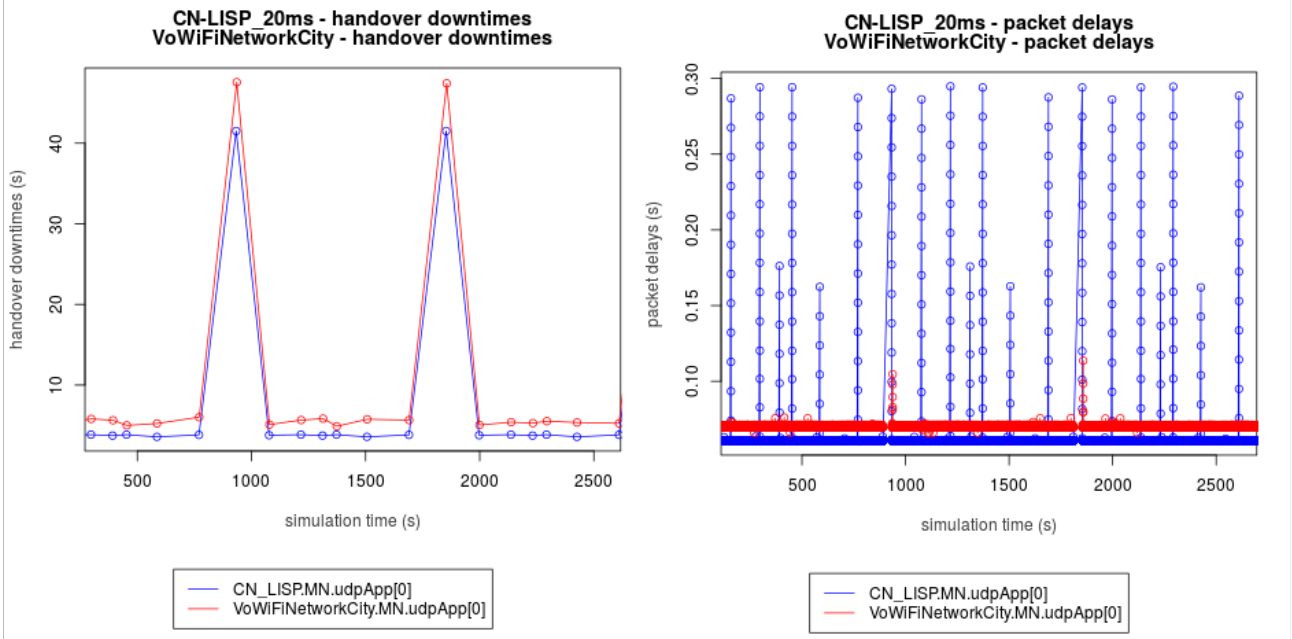
messageFreq = 20ms, *obstacles* = *no*



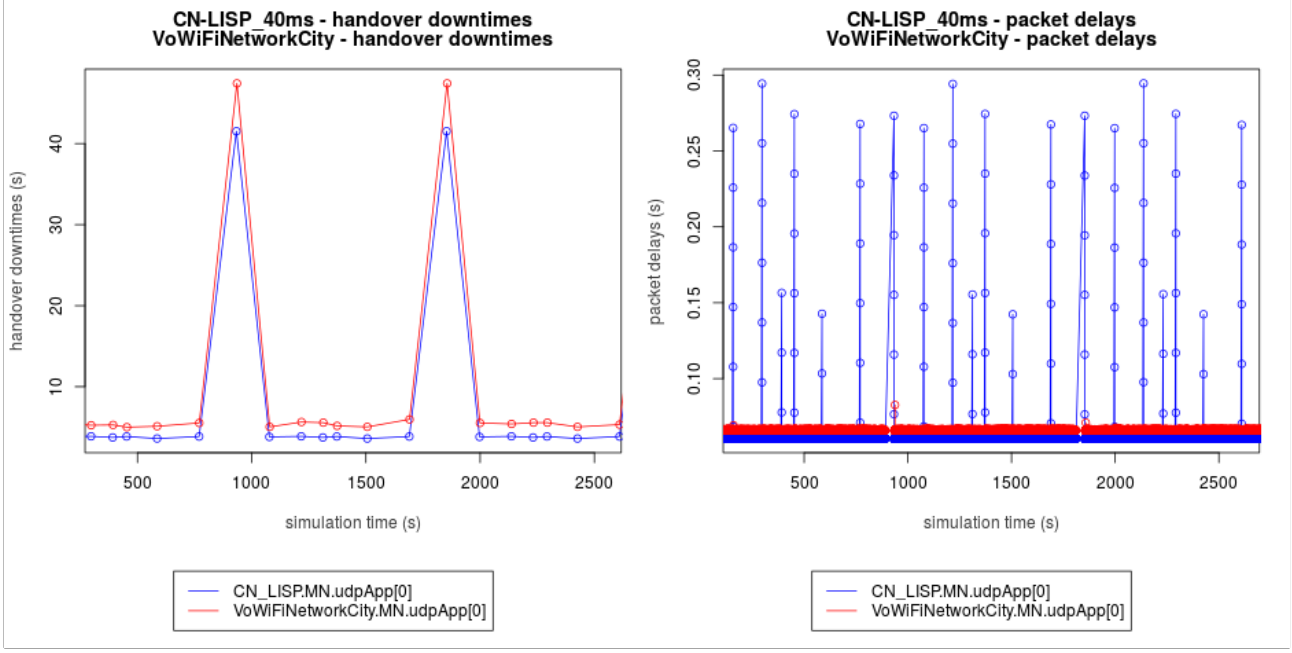


CN-LISP vs. MIPv6

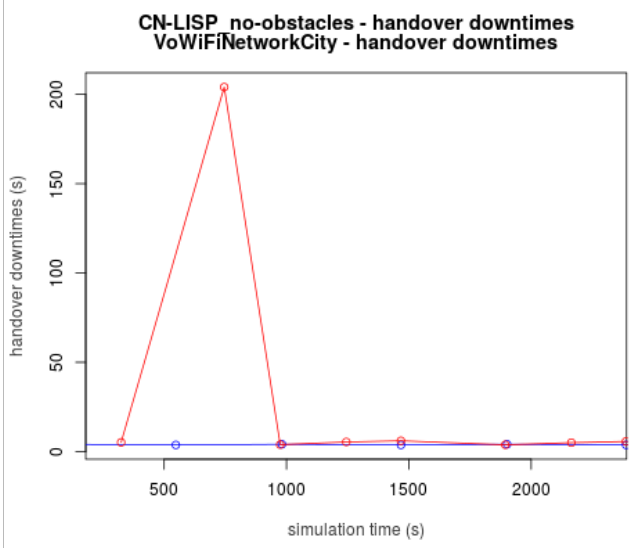
messageFreq = 20ms, *obstacles* = *yes*

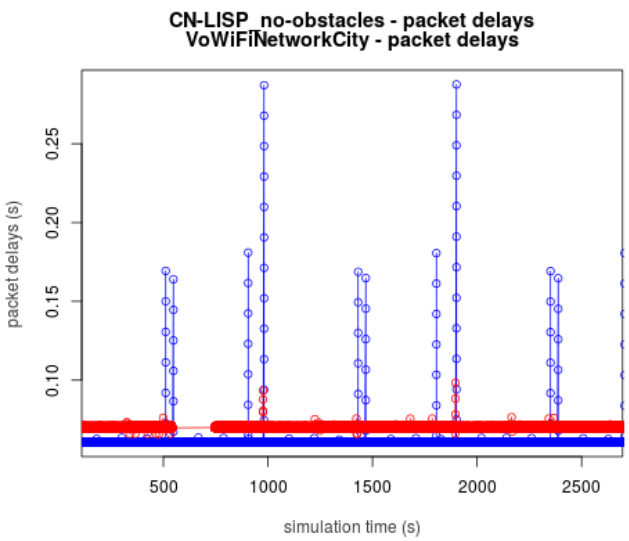


messageFreq = 40ms, *obstacles* = *yes*



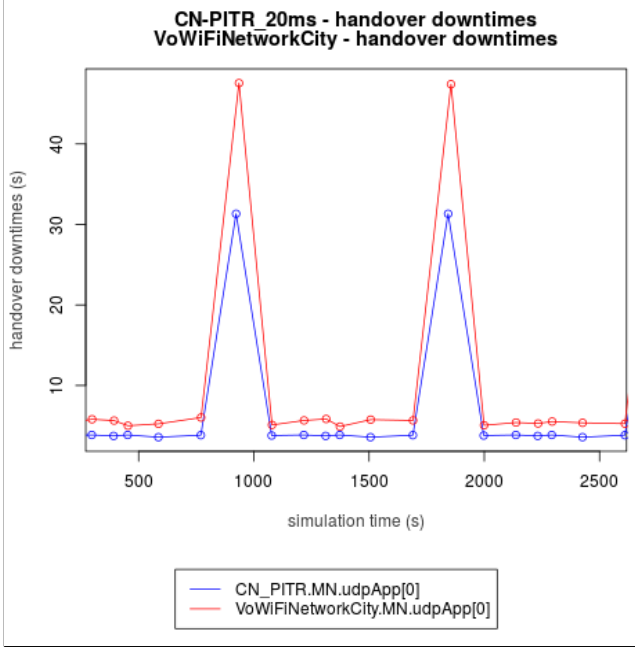
messageFreq = 20ms, *obstacles* = *no*

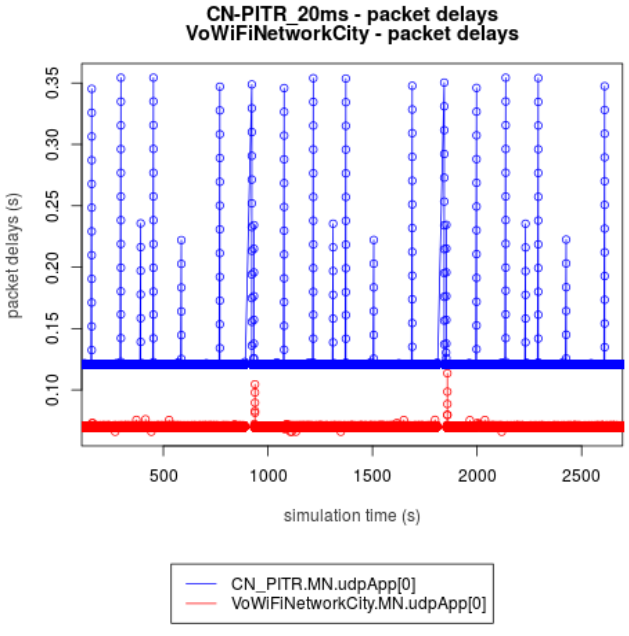




CN-PITR vs. MIPv6

messageFreq = 20ms, *obstacles* = *yes*





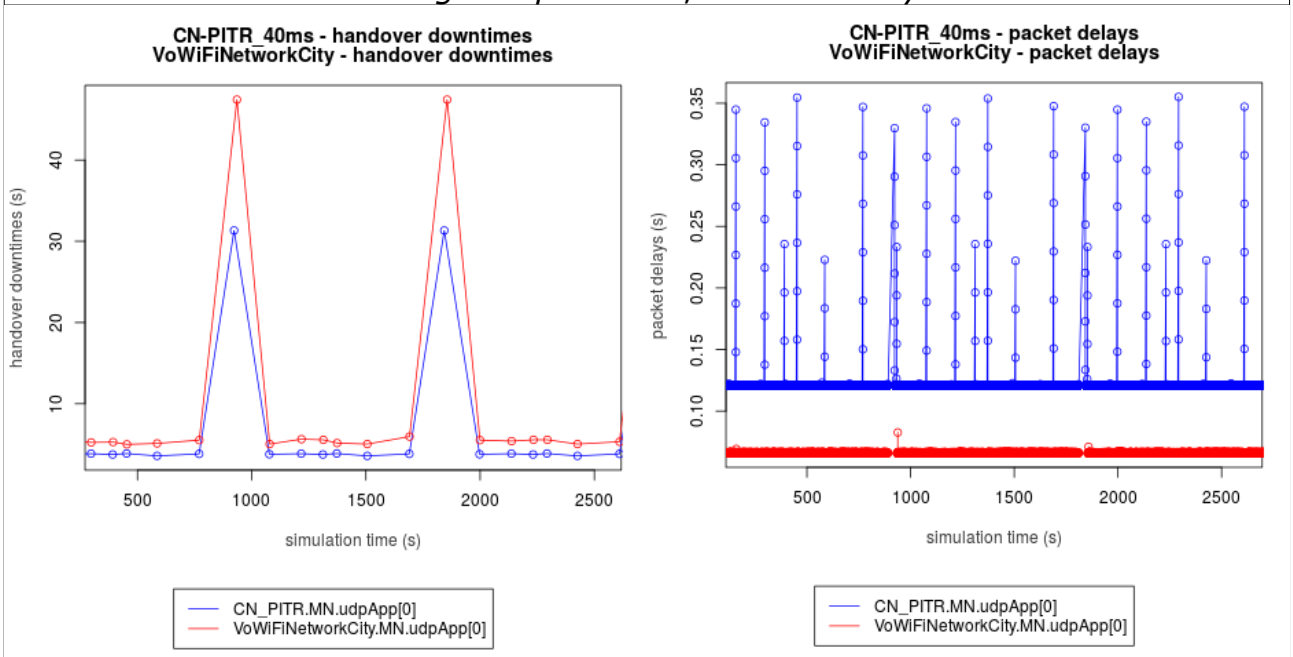
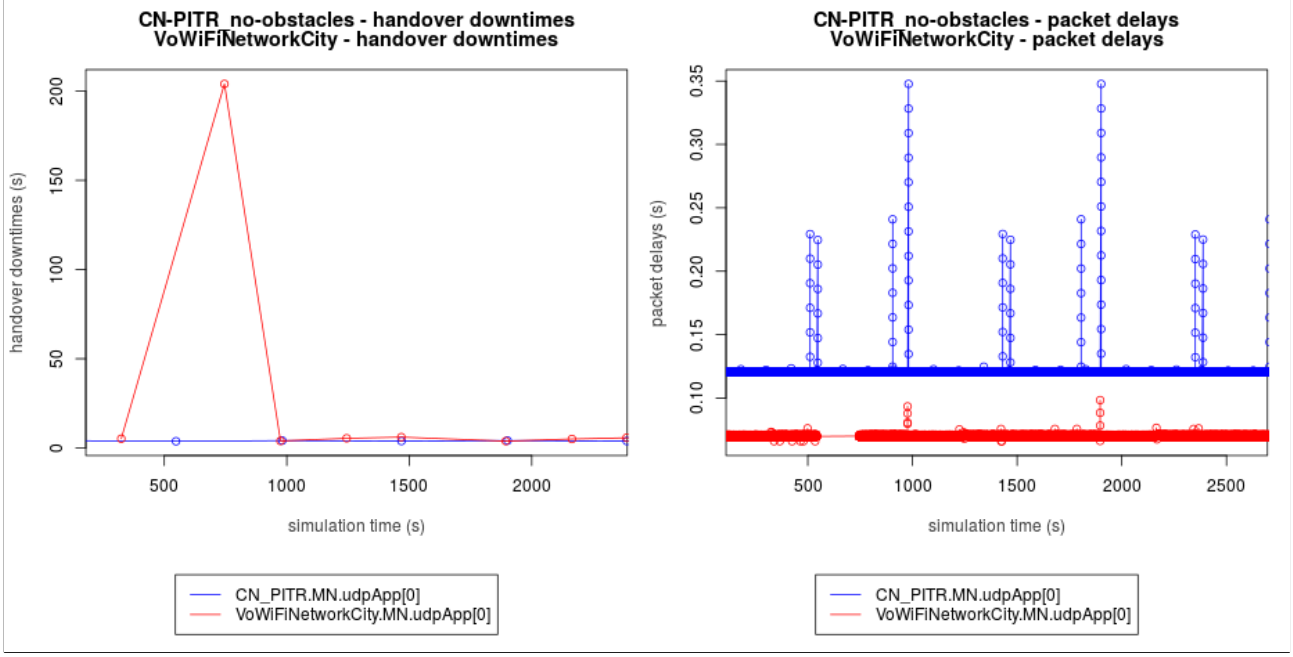
messageFreq = 40ms, *obstacles* = *yes*

messageFreq = 20ms, *obstacles* = *no*


Tabella 5

La Tabella 5 mostra i vari confronti in relazione alle due configurazioni di LISP. La prima considerazione nasce dal confronto delle due implementazioni, LISP e

ABPS, per i nodi mobili multihomed. Entrambe infatti prevedono un layer che permette di mascherare in maniera trasparente al livello sessione lo smistamento dei pacchetti tramite le interfacce di rete. Come evidenziato dai grafici precedenti, la proprietà del nodo di essere multihomed non sembra influenzare la latenza dei pacchetti durante la comunicazione; bensì ha una forte influenza sull'intervallo di indisponibilità, che dai ~1.3s di LISP senza utilizzo di più interfacce, scende sino ad un valore che oscilla tra i ~0.10s e ~0.30s; questo è possibile dal momento che l'altra interfaccia del nodo LISP, una volta configurata, registrerà il suo RLOC al Map Server ed avvierà la procedura di SMR. La durata dell'intervallo di indisponibilità di questi due protocolli, considerando la proprietà multihomed del nodo è dunque simile; le variazioni possono essere dovute a diversi fattori che non sono stati approfonditi in questo lavoro di tesi.

Per quanto riguarda la durata degli intervalli di indisponibilità per il confronto nella rete urbana, i grafici sono puramente indicativi di quanto, le comunicazioni mobili possano beneficiare dalla tecnologia e da una buona gestione del multihoming, in un ambiente come quello della mobilità. È però visibile dal confronto con MIPv6 come l'intervallo di indisponibilità introdotto da LISP sia inferiore di ~2.5s, sia nello scenario urbano che in quello test; questo può essere motivato da una migliore gestione dell'aggiornamento da parte di LISP, da una cattiva configurazione della rete MIPv6 o, ancora peggio, da una errata implementazione del simulatore stesso di MIPv6, che introduce anomalie nel comportamento durante l'handover.

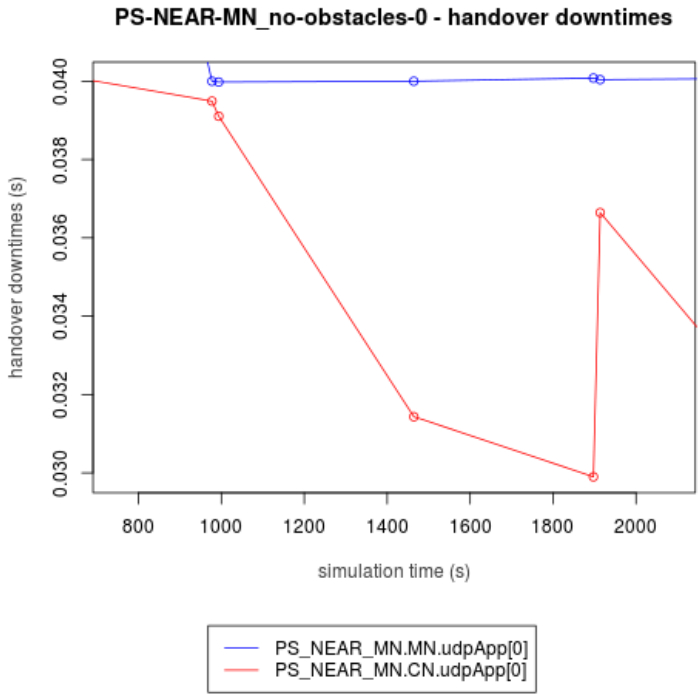
In termini di latenza dei pacchetti invece il discorso è limitato al percorso compiuto dai messaggi per giungere a destinazione. Nella rete urbana si nota chiaramente che, l'unica configurazione ad avere delle alte latenze é quella di LISP che sfrutta il Proxy ITR per la comunicazione con i siti non-LISP. La

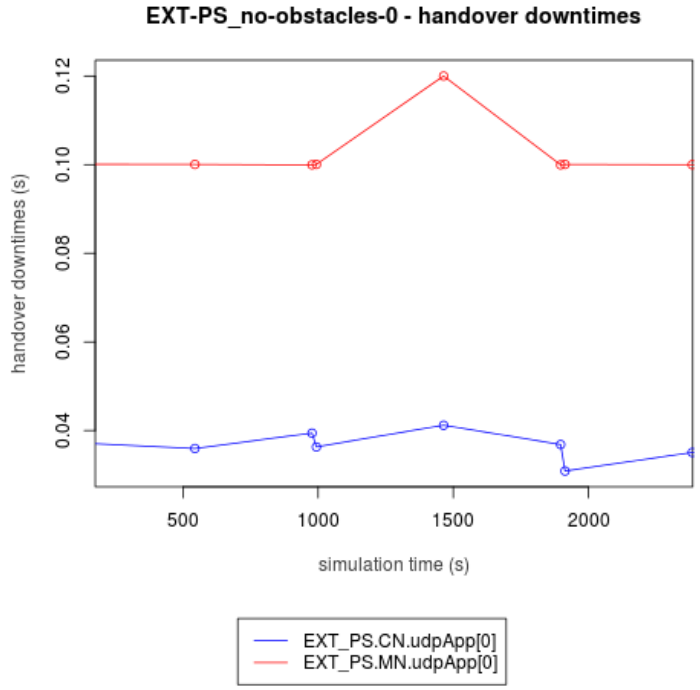
latenza dei pacchetti per le altre configurazioni corrisponde, nella maggior parte dei casi, al tempo impiegato per percorrere la backbone (due link da 30ms, quindi ~60ms); questo si verifica nello scenario urbano in quanto, per tutta la durata della simulazione, il nodo mobile non si allaccia mai alla rete in cui risiede il CN, dunque il flusso di pacchetti è costretto ad essere inoltrato attraverso i link ad alta latenza.

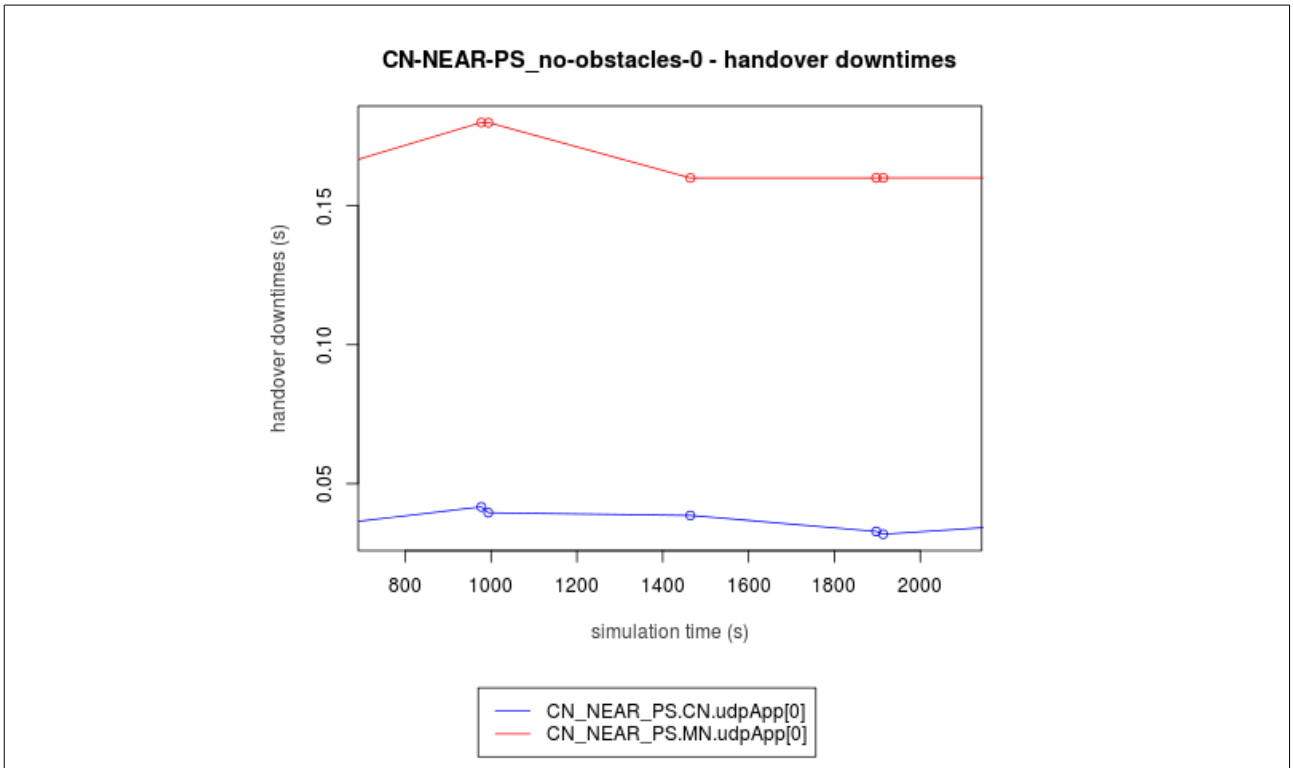
Nella precedente tabella è stata omessa la configurazione in cui il Proxy Server ABPS è posizionato nella stessa rete del nodo mobile; nei grafici infatti non si riesce a capire la variazione che questa avrebbe comportato in termini di intervallo di indisponibilità, in quanto resa nulla dai valori molto elevati del protocollo LISP con nodo mobile con singola interfaccia. Viene dunque fatta un'analisi di come l'intervallo di indisponibilità in ABPS dipenda dalla posizione del suo Proxy Server ABPS, e di come questo dipenda anche dalla frequenza dei messaggi dei due nodi comunicanti; mostrato in Tabella 6.

Scenario URBANO
PS-NEAR-MN vs. EXT-PS vs. CN-NEAR-PS

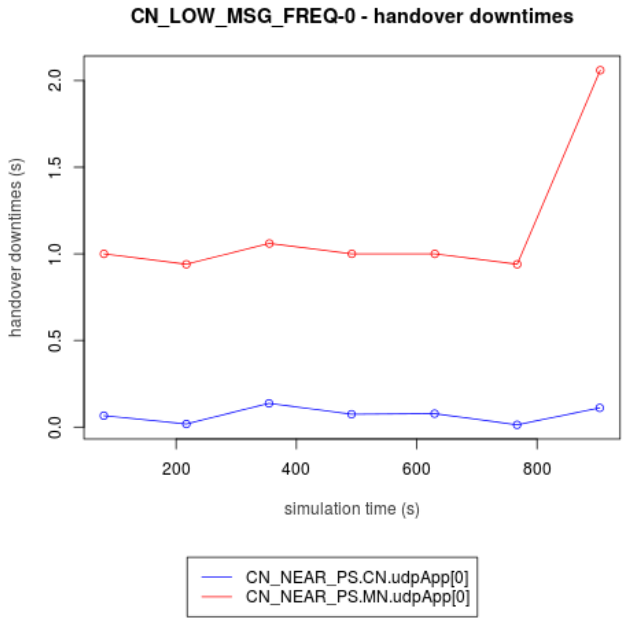
messageFreq = 20ms, *obstacles* = *no*







Scenario TEST EXT-PS	
CN.messageFreq = 1s	*MN.messageFreq* = 1s



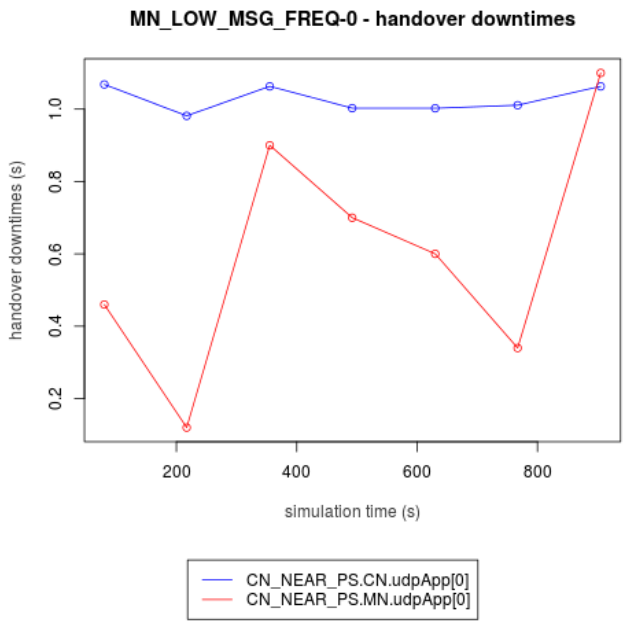


Tabella 6

La tabella sopra riportata mostra come l'intervallo di indisponibilità di ricezione, da parte del MN, sia direttamente proporzionale rispetto alla distanza del proxy server ABPS; che era un esito prevedibile, in quanto l'aggiornamento della posizione del nodo mobile avviene tramite di esso. Inoltre i grafici riguardanti il confronto della durata dell'handover, in casi in cui le due applicazioni abbiano frequeze di invio asimmetriche, evidenziano la necessità di introdurre dei meccanismi di probing; questi per permettere di effettuare l'update al proxy server ABPS ogni volta che il nodo mobile raggiunge una nuova rete e/o configura le proprie interfacce.

Capitolo 6

Conclusioni

6.1 Conclusioni

Ci sono due tipi di considerazioni da fare, una prima riguardante al lavoro svolto, quindi la creazione degli scenari e la valutazione dei protocolli, e una seconda riguardo la parte di correzione e introduzione di nuove componenti e caratteristiche ad entrambi i simulatori.

Per quanto riguarda la valutazione dei protocolli, questi si sono comportati come previsto (come mostrato in 5.4), in termini di latenza dei messaggi e tempo di handover, infatti con l'utilizzo concorrente di più interfacce questo si riduce notevolmente. È emerso che entrambe le architetture per il supporto alla mobilità presentano una componente mobile multihomed, che è in grado di ridurre in maniera drastica la durata dell'intervallo di indisponibilità. Dunque si può, come osservazione, precisare che ABPS limita il suo campo di azione alle comunicazioni VoIP tramite SIP ed RTP, mentre a livello teorico LISP è in grado di fornire i benefici della sua architettura a qualsiasi tipo di protocollo a livello sessione. ABPS, come punto di forza, fornisce una granularità per-packet che, data la mancanza nel simulatore di un'implementazione del layer di policies di QoS, non è stata studiata rispetto a LISP, dove questa componente è assente.

Inoltre il lavoro svolto ha portato i simulatori ad essere pronti per futuri sviluppi, approfondimenti e sperimentazioni; fornendo degli scenari su cui testare le metriche definite per la valutazione di architetture di supporto alla mobilità. Sono state introdotte diverse funzionalità e diverse correzioni alla

precedente implementazione (maggiormente di ABPS), che portava i simulatori a mostrare anomalie nel comportamento del protocollo, come descritto in 4.4.

6.2 Sviluppi futuri

Nonostante i simulatori trattati siano stati controllati, e si sia cercato di eliminare ogni comportamento anomalo che presentavano, è sempre possibile che qualche caso, o situazione particolare, sia sfuggita all'analisi effettuata. Dunque si consiglia di sperimentare nuovi scenari, provando configurazioni inusuali (es. diversi MN che comunicano con lo stesso CN per ABPS) e osservare, e nel caso correggere, eventuali malfunzionamenti. Inoltre sarebbe utile effettuare un porting alle versioni più recenti di INET per i due simulatori, e nel farlo eliminare (parlando del simulatore di ABPS) tutti i dettagli di implementazioni non necessari. Questo porting dovrebbe inoltre essere una libreria statica su OMNeT++, che a sua volta utilizza INET come libreria da estendere, in modo da non creare confusione tra le componenti native di INET e le componenti introdotte o modificate dai simulatori; trattando entrambi come due librerie separate dovrebbe essere successivamente più semplice mantenere aggiornato il codice dei simulatori con il procedere degli aggiornamenti su INET.

Si potrebbero analizzare nel dettaglio quali siano i benefici in termini di prestazioni della granularità per-packet fornita dall'architettura ABPS, e quanto il livello di Policies di QoS possa influire l'efficienza della comunicazione in relazione alla mobilità del terminale.
Il simulatore ABPS, nonostante sia stato aggiunto l'ultimo step del suo percorso (proxy server e CN), è ancora molto idealizzato e non risponde completamente alle specifiche definite in [7]. Dunque una possibile strada è quella di inserire i meccanismi di update, registrazione e instaurazione della chiamata VoIP, propri del protocollo ABPS; per fare ciò potrebbe essere

possibile dover prima pensare ad implementare a livello sessione i protocolli SIP e RTP. Infatti, in questo simulatore la comunicazione è identificata dal proxy client id anche durante lo scambio di dati, mentre in una situazione reale la comunicazione RTP è identificata dalla porta sulla quale è stata instaurata.

Il simulatore LISP, sebbene sia molto fedele per meccaniche alle specifiche descritte in [1], dovrebbe poter prevedere l'utilizzo di più Proxy ITR (almeno uno per ogni sito non-LISP); sono da rivedere le meccaniche in cui il nodo mobile aggiorna i mapping remoti di questi PITR, come specificato in 4.4.2.1 è definito un solo PITR da aggiornare. Dovrebbe essere implementato il sistema di mapping LISP+ALT, che permetterebbe di collocare più Map Server (che mantengono i mapping per uno o più siti LISP) permettendo la realizzazione di scenari più verosimili. Infine, la politica con cui è gestito lo scan dei canali radio e la configurazione degli indirizzi IP per le interfacce, crea dei problemi; quando un'interfaccia non riesce a rilevare comunicazioni sui suoi canali (o non riesca a configurarsi) notifica un errore generato dalla richiesta di un'ulteriore scansione mentre l'interfaccia sta già facendo scan. Le interfacce inoltre, in situazioni di multihoming si allacciano agli stessi access point, a meno che manualmente non si impostino canali diversi per ognuna di queste.

6.3 Bibliografia

[1] D. Farinacci, V. Fuller, D. Meyer, D. Lewis, Cisco Systems, "The Locator/ID Separation Protocol (LISP)", RFC 6830, January 2013.
[2] D. Farinacci, lispers.net, D. Lewis, cisco Systems, D. Meyer, 1-4-5.net, C. White, Logical Elegance, LLC., "LISP Mobile Node", draft-meyer-lisp-mn-14, January 7 2016.
[3] D. Farinacci, V. Fuller, Cisco Systems, "Locator/ID Separation Protocol (LISP) Map-Server Interface", RFC 6833, January 2013.
[4] D. Farinacci, V. Fuller, D. Meyer, D. Lewis, Cisco Systems, "Locator/ID

Separation Protocol Alternative Logical Topology (LISP+ALT)", RFC 6836, January 2013.
[5] D. Farinacci, V. Fuller, D. Meyer, D. Lewis, Cisco Systems, "Interworking between Locator/ID Separation Protocol (LISP) and Non-LISP Sites", RFC 6832, January 2013.
[6] D. Klein, M. Hoefling, M. Hartmann, M. Menth, "Integration of LISP and LISP-MN into INET", Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques, March 2012.
[7] V. Ghini, S. Ferretti, F. Panzieri, "The “Always Best Packet Switching” architecture for SIP-based mobile multimedia services", Journal of Systems and Software, vol. 84, no. 11, pp. 1827-1851, 2011.
[8] C. Perkins, Tellabs, D. Johnson, Rice University J. Arkko, Ericsson, "Mobility Support in IPv6", RFC 6275, July 2011
[9] L. Regazzi, "Simulazione del protocollo TCP a ritrasmissione asimmetrica anticipata su WiFi" (tesi di laurea), Dipartimento di Informatica - Scienza e Ingegneria, Università di Bologna, March 2014, Italia
[10] "OMNeT++", <https://omnetpp.org/intro>
[11] "INET Framework", <https://inet.omnetpp.org/Introduction.html>
[12] "Omnet Manual", <https://omnetpp.org/doc/omnetpp/manual/usman.html>

Appendice

A Installazione INET e simulatori

L'installazione di INET è abbastanza semplice, se si sta utilizzando quello preconfigurato per questo lavoro di tesi. Infatti è sufficiente importarlo nell'ambiente di OMNeT++ come progetto preesistente e compilarlo. Una volta effettuate queste operazioni, è pronto per essere utilizato. Se si trovano dei problemi di compilazione o simili, assicurarsi dalla configurazione della build, che venga utilizzata *src* ome unica cartella dei sorgenti. È fortemente consigliato utilizzare la versione 4.3.1 di OMNeT, in quanto, i due simulatori nelle versioni più recenti generano degli errori di compilazione che non ne permettono l'utilizzo.

B Componenti principali

Le parti della directory di INET, da noi utilizzate sono state: *src* e *simulations*. La prima contiene i sorgenti dell'implementazione di INET; la seconda è la cartella che contiene le varie sub-directory degli scenari simulativi configurati. La directory simulations è così strutturata:

- INET
  - simulations
    - <nome scenario>
      - analisys
      - plots
      - results
      - routing

      <file ned dello scenario>

      <file ini dello scenario>

La cartella *analisys* contiene, per comodità, tutti i file .anf, che permettono di analizzare ed osservare i risultati delle simulazioni contenuti nella cartella *results*. La directory *routing* raccoglie tutte le tabelle di routing .irt dei vari nodi che compongono la simulazione. Infine il file .ned e il file .ini vengono utilizzati rispettivamente per definire la rete sulla quale si vuole lavorare e per configurarla e definirne i parametri delle sue componenti. Per approfondimenti vedi [12].

C Accorgimenti

Ogni modifica che è stata apportata ad entrambi i simulatori è preceduta dal commento /* Convertino Sitta */, seguito da una breve spiegazione del perchè è stata effettuata. Per quanto riguarda i simulatori è opportuno specificare che, molti errori runtime che si dovessero presentare, dovuti al comando ASSERT presente nel codice sono spesso causati dal nodo mobile, il quale, o chi altri fa parte della comunicazione, non riesce a configurarsi per qualche ragione (es. non è riuscito ad assegnare un indirizzo IP alla sua interfaccia perchè non riceve il segnale dell'access point) prima che l'applicazione sui nodi cominci a inviare messaggi. Se non si risolvesse il problema, o ancora peggio, il programma si chiudesse causa segmentation fault, è consigliabile avviare la simulazione in modalità di DEBUG (integrata in OMNeT++), che permetterà di verificare manualmente quale comando ha generato il problema, e quali sono i valori delle variabili al momento del crash dell'applicazione. Nel caso si effettuassero altre simulazioni con gli scenari elaborati, si consiglia di impostare questi parametri nello scenario urbano:

```
**.wlan*.agent.minChannelTime = 0.10s
**.wlan*.agent.maxChannelTime = 0.12s
```