UNIVERSITÀ DEGLI STUDI DI PISA

Facoltà di Ingegneria

Corso di Laurea Specialistica in Ingegneria delle Telecomunicazioni

TESI DI LAUREA SPECIALISTICA

Progetto e sviluppo di una soluzione con DIAMETER per l'interazione tra Mobile IPv6 e la piattaforma AAA dell'operatore

Relatori

Prof. Ing. Stefano Giordano

Ing. Rosario G. Garroppo

Ing. Michele La Monaca

Candidato

Luca Battistoni

Alla mia famiglia

C'è al mondo una sola cosa
peggiore del far parlare di sé:
il non far parlare di sé.
*(*Oscar Wilde – *da* Il ritratto di Dorian Gray*)*

Ringraziamenti

Desidero ringraziare sentitamente i professori Stefano Giordano e Rosario Garroppo per i preziosi insegnamenti trasmessi con passione nei corsi da loro tenuti: è grazie a loro e agli altri componenti del TLCNetgroup del Dipartimento di Ingegneria dell'Informazione dell'Università di Pisa che ho iniziato a conoscere e ad amare il multiforme mondo delle reti di telecomunicazioni.

Nondimeno ringrazio l'Ing. Ivano Guardini che mi ha dato la possibilità di svolgere questa attività di tesi presso il Telecom Italia LAB di Torino: un'opportunità importante di crescita e di avvicinamento al mondo del lavoro, per la quale sono personalmente grato a lui ed all'azienda che rappresenta.

Un ringraziamento particolare va al mio tutor aziendale, Ing. Michele La Monaca: grazie per avermi accompagnato in quest'avventura, di avermi supportato (e sopportato) nei momenti di incertezza e difficoltà con esperienza e saggezza, e di avermi insegnato con professionalità come

lavora sul campo un ingegnere.

Un grazie va a tutti i colleghi incontrati in Tilab, da quelli con i quali ho condiviso il "caldo" laboratorio (Ivano, Alessandro, Carlo, Luca, Gennaro, Carla) a quelli dei "piani alti" (Davide, Marco, Patrick, Simone, Gerardo, Loris, Elena, Fabrizio): mi piace pensarvi tutti insieme in un'affollatissima pausa caffé del dopo pranzo…

E non posso dimenticare coloro che mi sono stati vicino in questi mesi di lavoro torinesi ma anche negli splendidi anni trascorsi tra La Spezia e Pisa: Luca, Pietro, Irene, Serena, Andrea, Michele, Paola, Elena, Arianna, Marco, Marco, Silvio, Alberto, Paolo, Luca, Giovanni, ma l'elenco è molto più lungo e non me ne vogliano tutti gli altri se li unisco tutti insieme in un corale: grazie!

Infine un ringraziamento alla mia famiglia e agli amici di sempre, del paese e del gruppo: persone che con la loro vicinanza e comprensione, specie nei momenti più difficili e stressanti, mi hanno infuso disinteressatamente la forza e il coraggio per portare a termine questo lungo e faticoso percorso. Non so cosa avrei fatto senza di voi, vi sono grato.

Indice

Introduzione

Negli ultimi anni, i terminali mobili hanno assunto un ruolo sempre più importante nelle reti di telecomunicazioni: con il frenetico sviluppo di nuove tecnologie e applicazioni, questo trend è oggi più attuale che mai.

Al giorno d'oggi, oggetti quali telefoni cellulari e computer portatili (notebook) sono ormai entrati a far parte dell'uso comune. Ma già da qualche tempo si sta assistitendo ad un fenomeno di convergenza di questi due "mondi" con la diffusione di smartphone e computer palmari/PDA. Tali terminali sembrano incontrare il favore degli utenti sia business che consumer e si propongono come i dispositivi portatili della nuova generazione.

Questi terminali sono sempre più potenti in quanto a capacità di calcolo, hanno memorie di massa e RAM sempre più grandi ed autonomie sempre maggiori. Al crescere della complessità di tali apparati, aumentano anche i requisti in termini di capacità trasmissiva (in mobilità) che gli operatori saranno tenuti a soddisfare nel prossimo futuro con diverse opzioni tecnologiche di accesso alla rete: UMTS, HSDPA, WiFi, Wi-Max, solo per citarne alcune.

Introduzione

Quantunque non si chiaro quale/i di queste tecnologia/e godranno del favore degli utenti per far fronte ai nuovi requisiti di banda, l'utilizzo della comunicazione a pacchetto basata sul protocollo IP non sembra temere rivali. Il protocollo IPv4 (l'attuale versione di IP) sarà sostituito nel prossimo futuro da una nuova versione (IPv6), per la quale è stato definito un protocollo ad-hoc per gestire la mobilità dei terminli (Mobile IPv6). Tale protocollo fornisce un piano di controllo e opportuni meccanismi per garantire la raggiungibilità dei terminali che cambiano il loro punto di accesso alla rete (situazione non rara per terminali mobili e multihomed).

L'infrastruttura di rete che realizza la mobilità è gestita tipicamente dagli operatori mobili, i quali necessitano di meccanismi di controllo dell'identità degli utenti che accedono alla loro rete, nonché delle loro autorizzazioni e dell'utilizzo fatto delle risorse di rete (per esigenze di tariffazione e/o di auditing). Per un operatore mobile, questo si traduce nella necessità di allestire una piattaforma di Autenticazione, Autorizzazione ed Accounting (AAA) per il controllo del servizio di mobilità.

Da ciò emerge la finalità di questo lavoro di tesi: lo studio e la prototipazione di una soluzione per l'integrazione degli apparati necessari alla realizzazione della mobilità in ambito IPv6 nell'infrastruttura AAA di un operatore. La soluzione proposta prevede l'impiego del protocollo AAA di nuova generazione: DIAMETER.

Il capitolo 1 descrive il protocollo Mobile IPv6, i componenti necessari e i meccanismi che permettono ai terminali mobili di essere raggiungibili quando si spostano da una rete IPv6 ad un'altra.

Il capitolo 2 affronta il problema della configurazione automatica

delle informazioni necessarie all'avvio (bootstrap) del servizio Mobile IPv6 e presenta le due soluzioni soluzioni proposte in normativa (applicabili a seconda dello scenario di roaming).

Nel capitolo 3 è descritto il protocollo DIAMETER assieme ad una panoramica dei componenti della piattaforma AAA.

Nel capitolo 4 è presentato un approccio tecnico per gestire l'interazione tra l'Home Agent (router che funge da anchor point nel protocollo Mobile IPv6) e il server AAA, ovvero l'interazione tra il mondo Mobile IPv6 e quello dell'autenticazione, autorizzazione ed accounting dell'operatore.

Infine il capitolo 5 descrive il lavoro di prototipazione sviluppato per l'implementazione della soluzione proposta ed il necessario trial sperimentale approntato.

Questo lavoro di tesi è stato realizzato presso i laboratori di Telecom Italia nella sede di Torino (TILab, ex Cselt).

Capitolo 1
Il protocollo Mobile IPv6

1.1 Introduzione

IPv6 (Internet Protocol version 6) [1] è un protocollo di rete (livello 3 della pila OSI) sviluppato in ambito IETF (Internet Engineering Task Force), la comunità aperta di tecnici, specialisti e ricercatori dediti all'evoluzione di Internet [2]. L'origine del protocollo si data all'inizio degli anni '90 allorché fu deciso di creare l'IPng (IP Next Generation) working group con l'obiettivo di realizzare un protocollo di rete di nuova generazione capace di risolvere in via definitiva la strutturale scarsità di indirizzi del suo predecessore (IPv4). In quegli anni tale problema veniva percepito come un nemico mortale per la nascente comunità di Internet. In realtà, alcune correzioni ed estensioni ad IPv4 (in primo luogo CIDR e NAT) ed una più accorta assegnazione degli indirizzi hanno permesso allo stesso di sopravvivere e prosperare a discapito del suo successore, tanto che

a oggi IP (alias IPv4) è l'indiscusso dominatore di Internet. Ciò nondimeno, nei prossimi anni IPv4 dovrà segnare il passo e cedere lo scettro ad IPv6 che, oltre ad un spazio di indirizzamento pressoché infinito, può vantare numerose migliorie rispetto al suo predecessore. Tra le sue caratteristiche, ricordiamo: il concetto di *visibilità* degli indirizzi (non più divisione tra indirizzi pubblici o privati ma passaggio ad indirizzi *scoped* in ambito link, global); un nuovo schema di indirizzamento con indirizzi di tipo unicast, multicast ed anycast; il formato degli indirizzi che passa da 32 a 128 bit per soddisfare il bisogno di indirizzi dovuto all'incremento vertiginoso del numero di utenti e dispositivi connessi ad Internet. Queste ed altre caratteristiche di IPv6 sono riprese e maggiormente dettagliate in Appendice A.

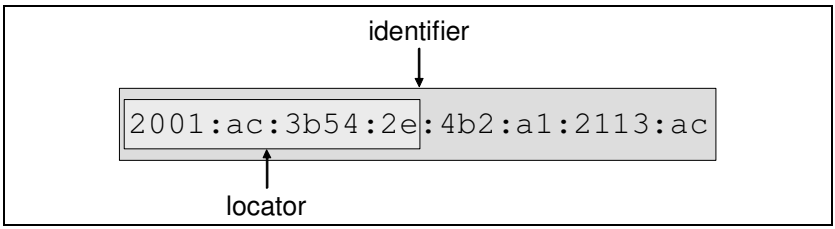

Figura 1.1 – Identifier vs locator

In IPv6 un indirizzo è suddivisibile in due parti [3]: i primi *n* bit identificano il prefisso di rete, i restanti 128-*n* bit l'indirizzo dell'interfaccia di rete. Questo significa che se un utente mobile (es. uno smartphone o un PDA) si sposta da una rete ad un'altra con un diverso prefisso, l'indirizzo globale al quale il nodo era precedentemente raggiungibile non è più

utilizzabile. Infatti l'indirizzo IP di un host è sia un identificatore con il quale può essere individuato e distinto dagli altri host della rete, sia un indicatore della posizione utilizzato dai router per instradare correttamente i pacchetti. Ovvero, in altri termini, un indirizzo IPv6, al pari di un indirizzo IPv4, è sia un *identifier* che un *locator*, come è esemplificato in Figura 1.1 dove abbiamo assunto che il prefisso di rete sia su 64 bit.

Per quanto detto, la mobilità e nomadicità dei nodi in reti differenti non è supportata nativamente in IPv6. Per far fronte a questa mancanza è stato sviluppato il protocollo Mobile IPv6 (MIPv6) [4] con lo scopo di consentire ai nodi IPv6 di spostarsi tra diverse reti (ovvero di poter cambiare il punto di accesso alla rete) con la garanzia di non vedere interrotte le sessioni TCP/UDP attive al livello superiore e di essere così sempre raggiungibili da parte degli altri nodi.

1.2 Panoramica del protocollo

Mobile IPv6 risolve il problema del dualismo identifier – locator assegnando ad un nodo mobile due indirizzi diversi per i due "ruoli".

Ogni nodo mobile (*Mobile Node* – MN) è univocamente e globalmente identificato dal suo *Home Address* (HoA); tale *identifier* è un indirizzo IPv6 relativo ad una rete, denominata Home Network, presso la quale il nodo risulta sempre raggiungibile (ad esempio l'intranet della Company A di Figura 1.2). L'HoA è staticamente associato al MN sia quando esso si trova nella sua Home Network sia quando esso si sposta in reti differenti, denominate, per contrasto, *Foreign Networks*.

Quando l'utente/nodo mobile migra in una Foreign Network, ad

esempio quando un impiegato della Company A si sposta con il suo notebook in trasferta presso la Company B, la rete ospitante fornisce un nuovo indirizzo IPv6 locale, denominato *Care-of Address* (CoA). Il CoA è il locator ed avrà necessariamente un diverso prefisso e che potrà essere configurato automaticamente sfruttando, ad esempio, il protocollo DHCPv6 [5].

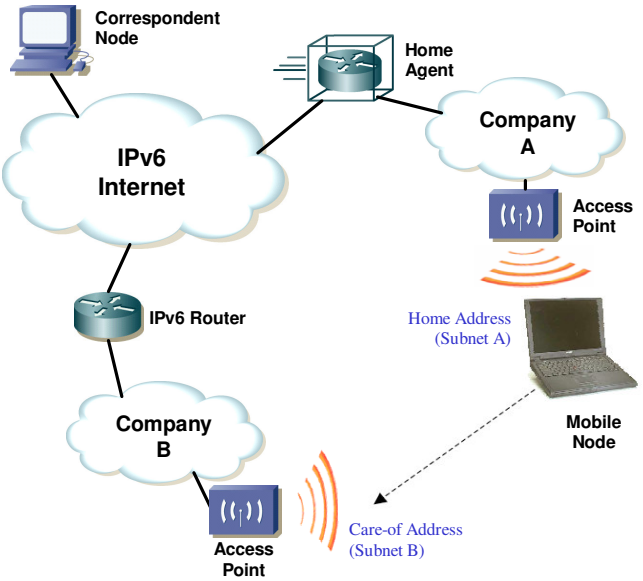

Figura 1.2 – Scenario di riferimento per Mobile IPv6

La raggiungibilità del MN presso la Company B è garantita dall'*Home Agent* (HA), router il cui compito è quello di intercettare i pacchetti destinati all'indirizzo Home Address del MN. In pratica l'HA

riceve i pacchetti destinati al MN e li inoltra al MN inviandoli al suo Care-of Address.

Affinché l'HA possa correttamente instradare il traffico verso il MN è necessario che esso conosca la corrispondenza Home Address / Care-of Address. comunicata attraverso un messaggio di Binding Update (BU – il formato del messaggio è descritto in Appendice B §B.1).

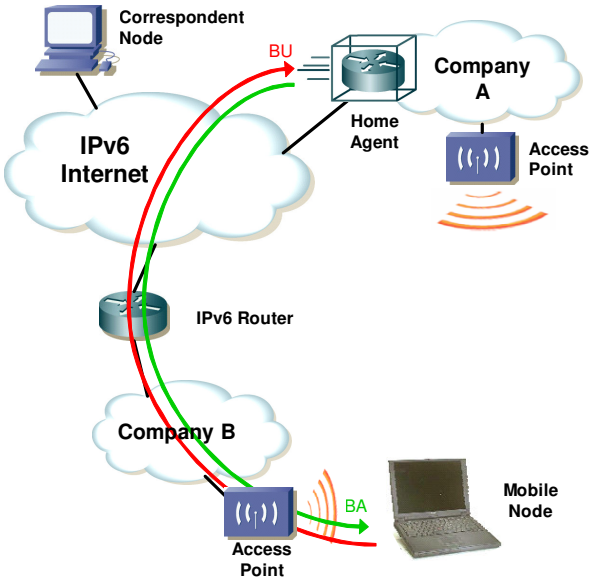

Figura 1.3 – Scambio di messaggi BU/BA

Una BU contiene nel campo Source Address il Care-of Address del MN mentre nella Home Address Option, una nuova opzione definita nell'RFC 3775 contenente l'Home Address del MN. L'Home Agent che

riceve una BU può così estrarre dal pacchetto IPv6 gli indirizzi HoA e CoA e creare il binding.

A conferma dell'avvenuta registrazione l'HA invia al MN un messaggio di Binding Acknowledgement (BA – il formato del messaggio è descritto in Appendice B §B.2). Una raffigurazione schematica di questa segnalazione è mostrata in Figura 1.3.

Per ovvi motivi di sicurezza, la segnalazione BU-BA deve essere protetta da meccanismi di sicurezza. Mobile IPv6 impone l'utilizzo di IPSec (IP Security), che sfrutta l'architettura di sicurezza di riferimento a livello IP per fornire meccanismi di riservatezza, autenticazione ed integrità tramite l'utilizzo degli extension header AH (Authentication Header) ed ESP (Encapsulated Security Payload) di IPv6. Ulteriori dettagli su queste caratteristiche di sicurezza sono presenti nel seguito (cfr. §1.3.6).

1.3 Descrizione dettagliata del protocollo

1.3.1 Bi-directional tunneling

Il generico nodo in comunicazione con un MN è definito Correspondent Node (CN). Abbiamo già accennato al fatto che l'Home Agent si fa carico di intercettare e instradare il traffico originato da un Correspondent Node che ha come destinatario un Home Address registrato presso di sé verso il Care-of Address corrispondente. A tal fine Home Agent e Mobile Node instaurano un tunnel IPv6-in-IPv6 bidirezionale (*Bi-directional Tunneling*).

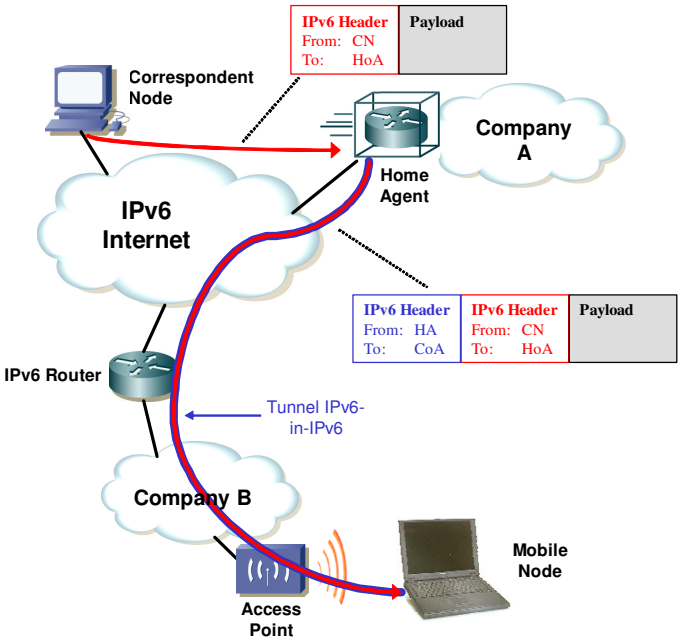

Figura 1.4 – Traffico CN→MN

Nella pratica l'Home Agent incapsula l'intero pacchetto IPv6 ricevuto da un CN in un nuovo pacchetto IPv6, il cui header ha nel campo Source Address l'indirizzo dell'HA e nel campo Destination Address il CoA del MN. Il MN utilizza un analogo meccanismo di tunneling per inviare i pacchetti al CN (*reverse tunneling*): inserisce il pacchetto IPv6 con il proprio HoA come sorgente e l'indirizzo del CN come destinatario in un ulteriore pacchetto IPv6 con sorgente il CoA e come destinatario l'HA, e spedisce il pacchetto a quest'ultimo. L'HA estrae il pacchetto incapsulato e lo inoltra al destinatario (CN). L'utilizzo del riverse tunneling è motivato

dal fatto che se il MN inviasse pacchetti direttamente al CN utilizzando l'HoA come sorgente il router di bordo della Company potrebbe scartare tali pacchetti, a causa del diverso prefisso dell'HoA.

In Figura 1.4 e Figura 1.5 riportiamo un esempio di comunicazione tra MN e CN che utilizza il bi-drectional tunneling.

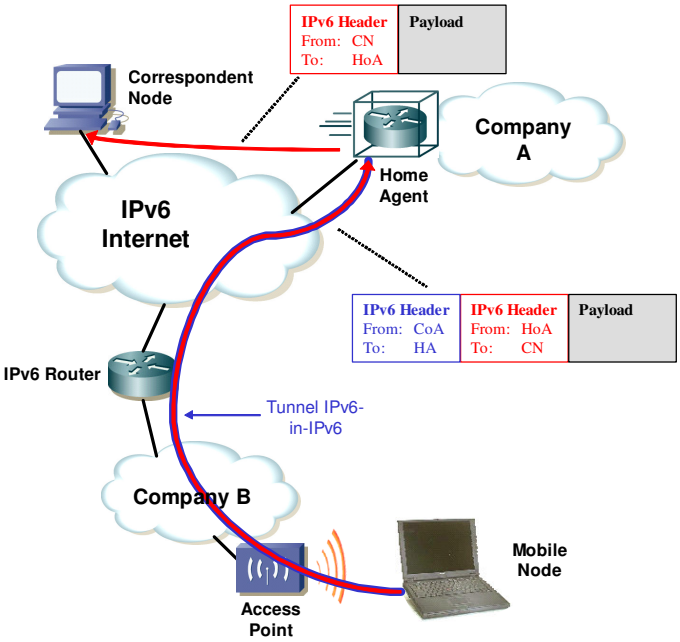

Figura 1.5 – Traffico MN→CN (reverse tunneling)

1.3.2 Route Optimization

La modalità di Reverse Tunneling non è però efficiente e può ridurre le prestazioni, in quanto tutto il traffico tra MN e CN è obbligato a

passare attraverso la Home Network e deve essere processato dall'HA. Meglio sarebbe se MN e CN potessero comunicare direttamente: questa funzionalità è prevista dallo standard e prende il nome di *Route Optimization*.

Per utilizzare la Route Optimization è necessario prima di tutto che tale caratteristica sia supportata dal CN e che, per motivi di sicurezza, il CN si assicuri preventivamente dell'effettiva raggiungibilità del MN sia al Care-of Address dichiarato che all'Home Address (cfr. *Return Routability Procedure* al sottoparagrafo successivo).

Se entrambe queste condizioni sono soddisfatte, il MN effettua uno scamio BU/BA con il CN che, al pari dell'HA, crea e mantiene localmente una corrispondenza HoA-CoA relativa al nodo mobile. Il CN può così inviare pacchetti direttamente al Care-of Address del MN. In realtà, il CN non può semplicemente inviare pacchetti IPv6 con il CoA del MN nel campo Destination Address, perché il MN si aspetta di ricevere pacchetti destinati al proprio Home Address.

Per ovviare a questo problema il CN aggiunge un routing header di "tipo 2" (il formato di questo nuovo header è descritto in Appendice B §B.4) nel quale è inserito l'Home Addess del MN. In questo modo il pacchetto, inoltrato dai router verso il Care-of Address del MN, potrà essere processato correttamente dal MN per la presenza del suo HoA nel routing header di tipo 2.

Riassumendo, la Route Optimization è sì una procedura opzionale più efficiente del (reverse) tunneling, ma è in generale sgradita agli operatori, in quanto il traffico degli utenti mobili non passa attraverso l'Home Agent e non è facilmente controllabile (ad esempio per

monitoraggio, shaping, tariffazione).

In Figura 1.6 presentiamo le due modalità di comunicazione tra MN e CN, bi-directional tunneling e route optimization, esemplificandole graficamente.

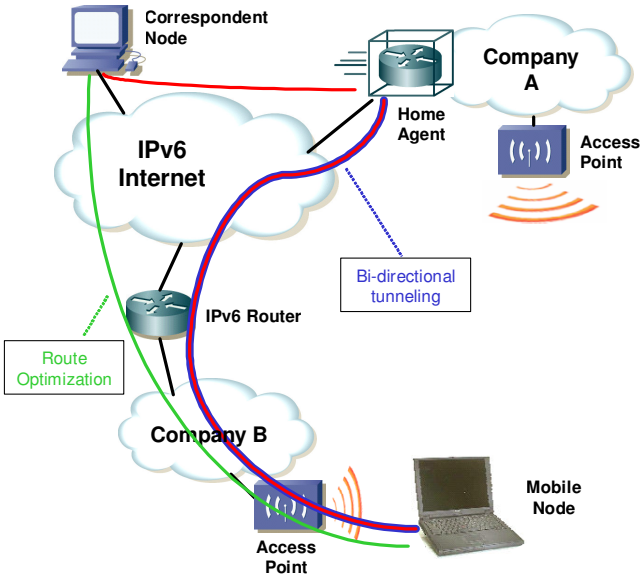

Figura 1.6 – Bidirectional tunneling vs. route optimizaion

1.3.3 Movement Detection

Affinché Mobile IPv6 funzioni è necessario un meccanismo, che prende il nome di *movement detection*, che permetta al MN di accorgersi del passaggio da una rete ad un'altra. ad una rete diversa.

Esistono diversi modi per un MN di preparare una movement detection. Nel seguito descriviamo una procedura di livello 3 che si basa

sulla verifica della raggiungibilità dei router utilizzati dal MN.

Un nodo mobile può accorgersi di avere cambiato rete attraverso i messaggi di Router Advertisement che i router della rete periodicamente inviano per pubblicizzare i prefissi IPv6 della rete medesima. Il fatto che il MN riceva l'informazione relativa ad un nuovo prefisso di rete, di per sé, non è sufficiente ad indicare lo spostamento: ad esempio potrebbe accadere che un nuovo router venga acceso nella rete alla quale è collegato il MN. In tal caso il MN può verificare la raggiungibilità del router in uso inviando un messaggio di Neighbor Solicitation: se il router interrogato non risponde con un Neighbor Advertisement il MN può realizzare che tale router non è più raggiungibile ed è necessario effettuare un handover di livello 3, ovvero il MN deve configurare un nuovo Care-of Address a partire dal prefisso pubblicizzato dal nuovo router di default.

1.3.4 Return routability

Prima di utilizzare la Route Optimization il CN deve comunque ottenere una certa sicurezza sulla raggiungibilità di un nodo mobile al suo Care-of Address dichiarato. Il test preventivo da compiere è noto come *Return Routability Procedure*. Esso prevede che il MN dia prova al CN di aver ricevuto una serie di dati, chiamati *keygen tokens*, a seguito di uno scambio di 4 messaggi:

- Home Test Init e Care-of Test Init, inviati dal MN al CN via HA il primo e direttamente il secondo, nei quali sono utilizzati rispettivamente il proprio Home Address e il Care-of Address come indirizzo sorgente, un Home Init cookie e un Care-of Init cookie e il destinatario è per entrambi il CN;

- Home Test e Care-of Test, inviati dal CN al MN su percorsi simili, nei quali sono trasmessi rispettivamente all'Home Address e al Care-of Address i cookie ricevuti, i token generati per i due messaggi e i nonce utilizzati (si usa una funzione di hash che omettiamo per brevità).

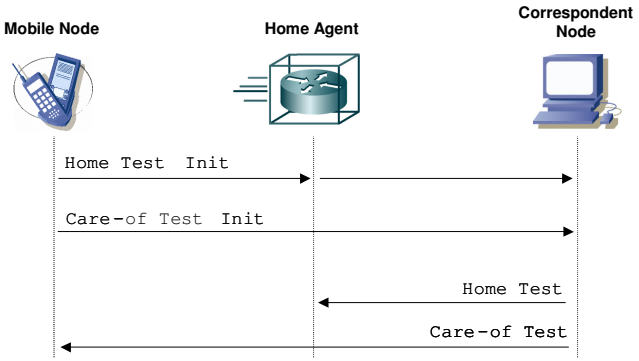

Figura 1.7 – Scambio messaggi per Return Routability Procedure

Concatenando i token ricevuti e facendone l'hash, il MN ottiene una chiave che utilizza per inviare la Binding Update al CN.

1.3.5 Strutture dati e nuovi messaggi

Gli Home Agent e i Correspondent Node che implementano la Route Optimization gestiscono localmente una struttura dati, la **Binding Cache**, ovvero una lista di entry delle corrispondenze HoA-CoA di utenti mobili. Nella Binding Cache può essere presente al più una entry per ogni Home Address ed ogni entry contiene un valore di lifetime ad indicare la

validità residua dell'entry stessa.

Ogni nodo mobile ha una **Binding Update List** che registra le informazioni relative a ciascuna binding che il MN ha o sta cercando di instaurare con un Home Agent o con un Correspondent Node.

I messaggi relativi al protocollo Mobile IPv6 sono inseriti dentro pacchetti IPv6 e sono preceduti da un header apposito, il **Mobility Header**. Esso è un extension header di IPv6 (per la definizione di extension header si veda in Appendice A il §A.3) identificato dal valore Next Header 135 ed è utilizzato dai nodi mobili, dai CN e dagli HA per la segnalazione relativa alla creazione e gestione delle corrispondenze HoA-CoA. Il formato è rappresentato in Figura 1.8.

Payload Proto identifica il tipo di header che segue il Mobility Header, utilizzando gli stessi valori del campo IPv6 Next Header; progettato per usi futuri, al momento è posto al valore IPPROTO_NONE (59 decimale).

Header Length rappresenta la lunghezza del Mobility Header in unità di 8 ottetti escludendone i primi 8: la lunghezza del Mobility Header deve essere un multiplo intero di 8 ottetti.

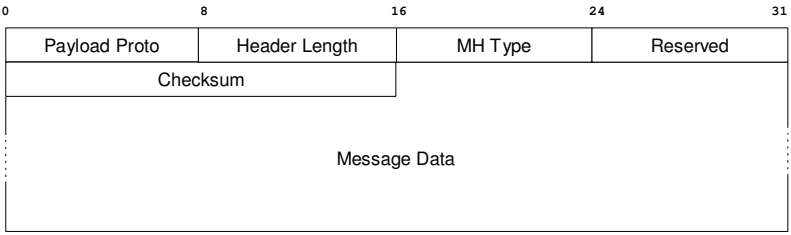

Figura 1.8 – Formato del Mobility Header

Nel campo *Message Data* è presente il particolare Mobility Message indicato dal valore del campo *MH Type* ed eventuali Mobility Options.

In Appendice B sono dettagliati i messaggi più importanti utilizzati nella segnalazione di Mobile IPv6 (Binding Update, Binding Acknowledgement, Binding Error), il Routing Header type 2 e la Home Address option..

1.3.6 IPsec e Authentication Protocol

Abbiamo già accennato nell'introdurre Mobile IPv6 che il protocollo prevede l'utilizzo di IPSec per rendere sicuro lo scambio di messaggi BU/BA tra nodo mobile ed Home Agent. Per i dettagli su IPSec e l'instaurazione di una Security Association (SA) rimandiamo all'Appendice A §A.6. Qui invece vorremmo accennare brevemente ad alcuni problemi circa l'utilizzo combinato di IPSec e Mobile IPv6.

Un primo problema generale, non specifico di IPsec, è la necessità di configurare i due apparati che utilizzano IPSec con il materiale crittografico opportuno. Questo può essere risolto utilizzando un'infrastruttura a chiave pubblica (Public Key Infrastructure – PKI) oppure protocolli di scambio di chiavi end-to-end quali IKEv2 [6]. Il prezzo da pagare è l'aumento della latenza nella registrazione del MN presso l'HA a causa dello scambio di messaggi necessari al recupero/negoziazione delle chiavi.

Inoltre l'utilizzo di IPSec tra MN ed HA può risultare problematico, se non impossibile, in caso di presenza di firewall nel path MN-HA.

Infine IPSec potrebbe risultare particolarmente gravoso in termini di potenza di calcolo per i terminali mobili (telefoni cellulari, smartphone, PDA).

- Un'alternativa più agevole per l'autenticazione (non cifratura) della segnalazione BU/BA tra MN ed HA è l'**Authentication Protocol**, descritto nell'RFC 4285 [7]. L'Authentication Protocol prevede l'utilizzo di una `Mobility Message Authentication Option` inserita nel campo Mobility Options dei messaggi BU/BA e che è utilizzata per autenticare tali messaggi. Il dettaglio della Mobility Message Authentication Option è presente nel §B.6 in Appendice B.

Capitolo 2
MIPv6 Bootstrapping

2.1 Introduzione

Nel capitolo precedente abbiamo visto come con Mobile IPv6 sia possibile mantenere la raggiungibilità di nodi mobili i quali, muovendosi, cambiano il loro punto di accesso alla rete. Un problema non trascurabile e non affrontato dal protocollo Mobile IPv6 è la definizione di meccanismi di configurazione automatica, nei nodi mobili, delle informazioni necessarie all'avvio del servizio di mobilità.

La configurazione manuale degli indirizzi non è una soluzione realisticamente praticabile per il deployment di un servizio di mobilità basato su Mobile IPv6 da parte di un operatore. Una procedura dinamica di configurazione è dunque imprescindibile per la scalabilità del servizio, in previsione di una notevole diffusione di dispositivi mobili always-on dotati di connettività dati, quali smartphone, PDA e UMPC. Tale procedura, inoltre, permetterebbe agli operatori di gestire in maniera più efficiente le

risorse, in termini di apparati di rete (Home Agents) e banda con un'accorta allocazione dell'HA in fase di bootstrapping.

Tale problematica, fortemente sentita dagli operatori, è descritta dettagliatamente in [8] ed ha spinto il working group IETF che si occupa di Mobile IPv6 (MIP6 wg) ad interessarsi del problema e cercare di porvi rimedio.

In [8] è chiarito che cosa si intenda per **bootstrap** di Mobile IPv6: il bootstrap è un meccanismo che fornisce al nodo mobile le informazioni necessarie all'avvio del servizio Mobile IPv6. Il processo di bootstrap tipicamente avviene all'accensione di un terminale mobile, ma può anche verificarsi quando un nodo mobile è riavviato in seguito ad un malfunzionamento senza aver conservato i dati ottenuti da un precedente bootstrap.

Il gruppo di lavoro di MIP6 ha definito il set minimo di informazioni necessarie per il bootstrap, che nella RFC 3775 [4] erano considerate preconfigurate nei nodi mobili:

1. l'indirizzo IPv6 dell'Home Agent;
2. l'Home Address del nodo mobile;
3. le credenziali per creare un'associazione di sicurezza IPSec tra MN e HA.

Qualsiasi soluzione per il bootstrapping non può prescindere da meccanismi di Autenticazione, Autorizzazione ed Accounting (AAA): come minimo, infatti, la piattaforma AAA dell'operatore deve verificare l'identità dell'utente mobile prima di fornire le informazioni per il bootstrap del servizio Mobile IPv6.

2.1.1 Terminologia

Spieghiamo di seguito il significato di alcuni acronimi e la funzione dei componenti dei quali parleremo diffusamente nel seguito del capitolo.

Le entità di business coinvolte in un ipotetico servizio basato su Mobile IPv6 sono gli operatori che forniscono l'accesso alla rete (ad esempio un Internet Service Provider oppure una rete intranet aziendale) e gli operatori che forniscono il servizio di mobilità, denominati rispettivamente Access Service Provider (ASP) e Mobility Service Provider (MSP).

I provider che fornisco il servizo possono non essere coloro che effettivamente lo autorizzano. Ad esempio, in uno scenario di roaming, l'ASP deve contatattare "l'Home ASP" dell'utente prima di autorizzarlo all'accesso; in tal caso si e soliti descrivere l'Home ASP come Access Service Authorizer (ASA). Parimenti, è ipotizzabile che in alcuni scenari il servizio di mobilità possa essere fornito da un entità con non sia l'"Home MSP" dell'utente. Analogamente in questo caso si parla di Mobility Service Authorizer (MSA)..

Sebbene siano l'ASA e il MSA ad autorizzare un utente all'accesso e alla mobilità rispettivamente, gli effettivi provider, ASP e MSP, possono applicare delle proprie politiche per limitare le autorizzazioni concesse agli utenti dai rispettivi Home Provider. Ad esempio, l'ASP può limitare un utente autorizzato all'accesso dall'ASA bloccando certe tipologie di traffico (ad esempio quello peer-to-peer).

Sono possibili differenti scenari a seconda che ASP, MSP, ASA e MSA siano o meno lo stesso operatore, con i casi limite

1) ASP=MSP=ASA=MSA;

2) quando le quattro entità sono quattro operatori diversi (caso più generale possibile), come rappresentato schematicamente in Figura 2.1.

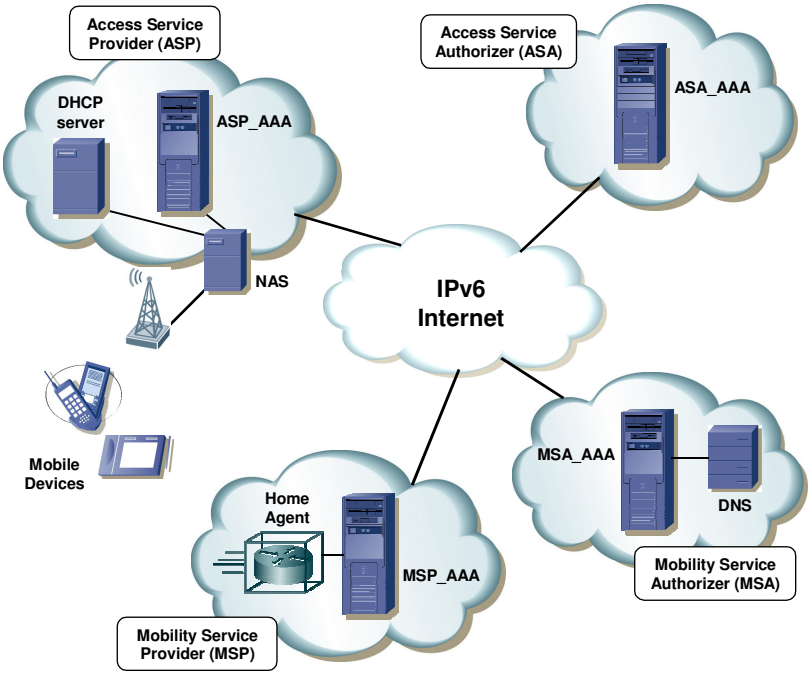

Figura 2.1 – Entità coinvolte nel bootstrap e nella gestione di Mobile IPv6

2.2 Gli scenari

Come descritto del paragrafo precedente, a seconda delle relazioni che intercorrono tra le entità ASP, MSP, ASA e MSA si possono

individuare molteplici scenari. Tra di essi due "macro" scenari rivestono un'importanza particolare perché ad essi si applicano soluzioni di bootstrapping differenti.

2.2.1 Split scenario

Lo scenario in cui Access Service Authorizer e Mobility Service Authorizer sono entità distinte viene definito **split scenario** [9]. In base alla relazione che intercorre tra MSA e MSP sono possibili due sotto-scenari:

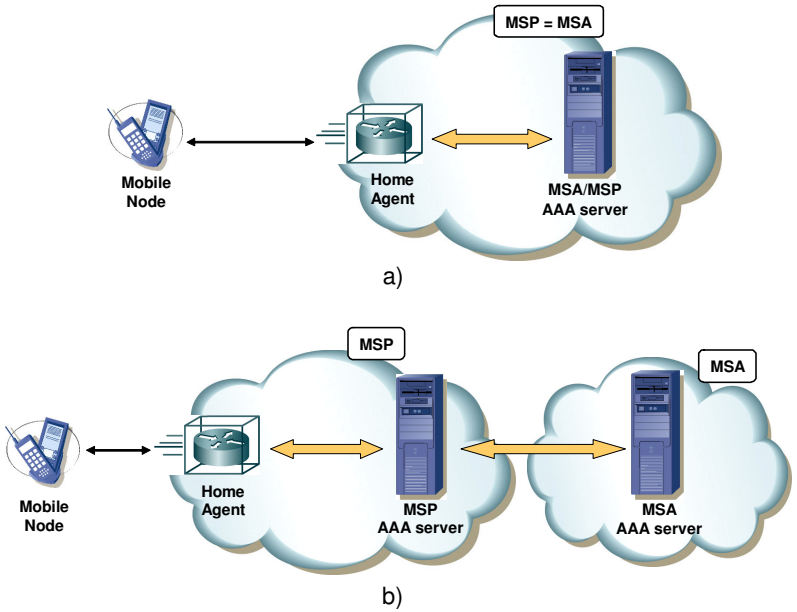

Figura 2.2 - Split scenario

- MSP e MSA sono la medesima entità (Figura 2.2-a);

- MSP e MSA sono entità diverse (Figura 2.2-b).

Le frecce in giallo indicano un'interfaccia di tipo AAA (si vedano Capitolo 3 e 4).

È inoltre possibile che l'utente mobile si trovi connesso ad una rete d'accesso diversa da quella alla quale è sottoscritto, ovvero sia in roaming: questo implica che l'ASP utilizzato (definito *serving ASP*) debba contattare per l'autorizzazione all'accesso il server AAA nella Home network dell'ASA. Ma questa possibilità vale sia per lo split che per l'integrated scenario, e non pregiudica la trattazione degli scenari medesimi, a meno di notare il fatto che ASP e ASA se gestiti da operatori diversi devono aver sottoscritto accordi di roaming.

2.2.2 Integrated scenario

Per **integrated scenario** si intende, come suggerisce il termine, lo scenario in cui ASA e MSA sono lo stesso operatore. Questo è lo scenario tipico in cui un utente sottoscrive un contratto con un operatore di telefonia mobile (TIM, Vodafone, Wind, etc.).

In questo scenario ASA e MSA possono intendersi come un'unica entità: quando anche ASP e MSP appartengono al medesimo operatore, si parla in letteratura di Integrated ASP. In generale l'ASP può essere separato dal ASA/MSA, ma anche in questo caso, ciò non influenza la trattazione dell'integrated scenario e pertanto nel seguito verrà spesso ripreso il caso Integrated ASP senza perdere di generalità.

Sfruttando la coincidenza di ASA e MSA, nell'integrated scenario è possibile fornire i parametri per il bootstrap durante la fase di accesso alla rete, contestualmente all'autenticazione e all'autorizzazione all'accesso.

In Figura 2.3 sono raffigurati gli elementi coinvolti in uno scenario integrated. Il Network Access Service (NAS) è l'apparato che fornisce al MN l'accesso alla rete previa interrogazione dell'AAA relativo (AAA_NA – AAA for Network Access [8]).

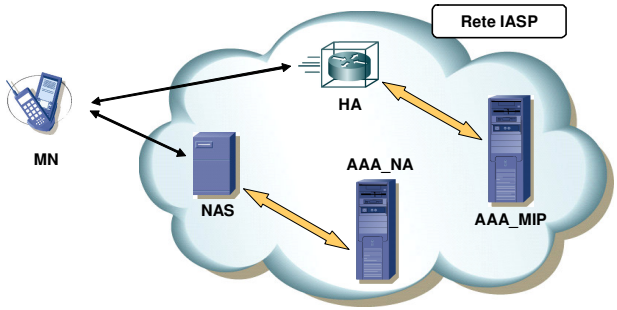

Figura 2.3 – Integrated scenario (ASP/MSP/MASA)

2.3 Soluzioni per il bootstrap di Mobile IPv6

Come spiegato nell'introduzione, il bootstrap di Mobile IPv6 comprende quattro fasi:

1. la scoperta dell'indirizzo dell'HA;
2. il setup della IPSEC Security Association (SA);
3. l'assegnazione dell'Home Address;
4. l'autenticazione ed autorizzazione con il MSA.

A parte il primo punto, ovvero la scoperta dell'indirizzo dell'Home Agent, diverso per split ed integrated, il resto della soluzione è comune ai due scenari. Le soluzioni per il bootstrap sono definite nei draft [9] e [10]

per gli scenari split ed integrated rispettivamente.

2.3.1 Scoperta dell'indirizzo dell'HA

2.3.1.1. Integrated scenario

Nell'integrated scenario il MN recupera l'indirizzo dell'HA tramite una richiesta DHCPv6. Dynamic Host Configuration Protocol version 6 (DHCPv6) [5], analogamente al suo predecessore DHCP, è il protocollo per la configurazioni dinamica di host IPv6. DHCPv6 prevede anche la presenza di agenti Relay per inoltrare le richieste di un host che non raggiunga direttamente un server DHCPv6. In questa soluzione è necessario richiedere che il NAS e il DHCP Relay Agent siano co-locati (cfr. Figura 2.4).

Descriviamo di seguito il meccanismo di assegnazione dell'HA mediante DHCPv6.

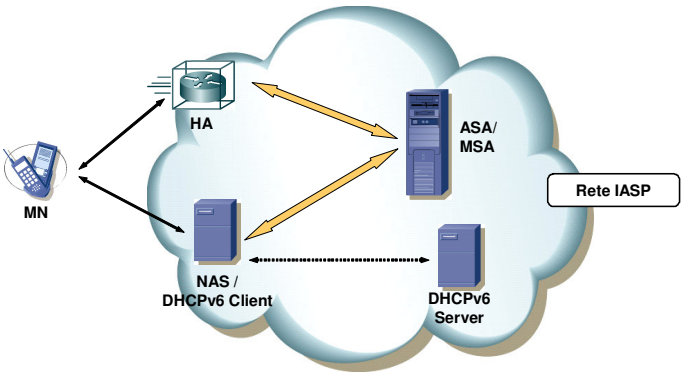

Figura 2.4 – Componenti per il bootstrap nell'integrated scenario

Il MN (step 1 in Figura 2.5) tramite il NAS ottiene accesso alla rete a seguito di un'interrogazione del server AAA del MSA/ASA. L'informazione dell'indirizzo HA viene inserita dal server AAA e memorizzata dal NAS/DHCP Relay.

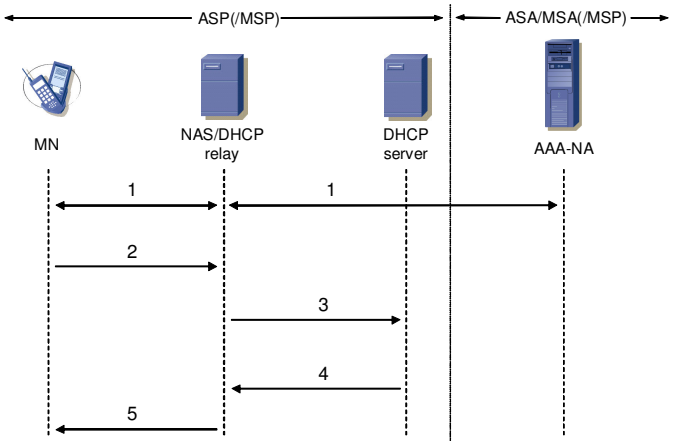

Figura 2.5 – Scambio di messaggi per la scoperta dell'Home Agent via DHCPv6

Una volta ottenuto l'accesso, il MN (step 2) invia un messaggio DHCPv6 di tipo Information Request all'indirizzo multicast corrispondente a tutti i server e relay DHCP richiedendo l'indirizzo dell'HA da utilizzare per il servizio Mobile IPv6. Questa richiesta arriva al NAS/DHCP Relay che la inoltra al DHCP server (step 3) aggiungendo in un'opzione (OPTION_MIP6-RELAY-Option [11]) l'indirizzo dell'HA ricevuto in precedenza dal ASA/MSA. Il server DHCP elabora la richiesta e invia la relativa risposta (step 4) nella quale inserisce l'indirizzo dell'HA

comunicato dal NAS. Questo messaggio è inoltrato dal NAS/DHCP Relay e giunge infine al MN (step 5), il quale è così in grado di contattare l'HA per avviare il servizio Mobile IPv6.

2.3.1.2. Split scenario

Nello split scenario non è possibile riproporre la soluzione integrated per la scoperta dell'indirizzo dell'HA. Questo perché ASA ed MSA appartengono ad operatori diversi e il NAS non può recuperare l'indirizzo dell'HA contestualmente all'autenticazione/autorizzazione per l'accesso presso il server ASA.

La scoperta dell'Home Agent nello split scenario viene fatta con una interrogazione al Domain Name System (DNS). La richiesta al DNS è relativa al Fully Qualified Domain Name (FQDN) dell'Home Agent (che deve essere quindi preconfigurato): a seguito di tale query il MN riceve in risposta l'indirizzo IPv6 (unicast oppure anycast) dell'Home Agent.

Alternativamente, la richiesta al DNS può essere fatta con un record di tipo SRV contenente con il nome del servizio "mip6. In risposta il MN otterrà gli FQDN di uno o più HA o, quando possibile, direttamente i loro indirizzi IP, evitando così ulteriori round trip per la risoluzione ricorsiva degli FQDN.

2.3.2 Setup della IPSEC Security Association (SA)

Il setup delle Security Association IPsec è effettuato tramite il protocollo IKEv2 [6]. Lo scambio di messaggi IKEv2 inizia tra MN e HA dopo la scoperta da parte del MN dell'indirizzo dell'HA. Per i dettagli di questa procedura si rimanda a [12].

2.3.3 Assegnazione dell'Home Address

L'altro parametro necessario al bootstrap di Mobile IPv6 è l'Home Address. Esso viene assegnato dall'HA durante la fase di autenticazione IKEv2 tra MN ed HA oppure semplicemente autoconfigurato dal MN (sempre durante lo scambio IKEv2).

Nel primo caso il MN richiede un'Home Address all'HA inserendo l'attributo INTERNAL_IP6_ADDRESS attraverso una CFG_REQUEST nella fase di IKE_AUTH (step 3 di Figura 2.6); l'HA processa il messaggio, alloca un HoA per il MN e lo comunica attraverso la CFG_REPLY (step 4) del messaggio di risposta. Per la procedura di autoconfigurazione rimandiamo per brevità a [9].

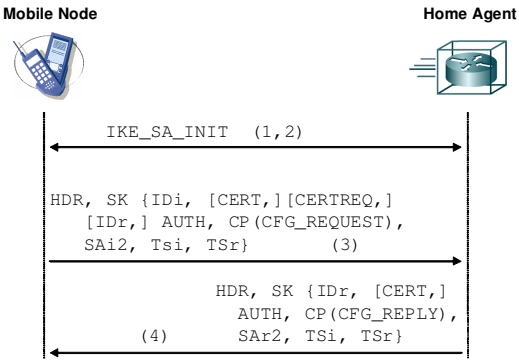

Figura 2.6 – Esempio di scambio di messaggi IKEv2 per l'assegnazione di HoA

2.3.4 Autenticazione ed autorizzazione presso MSA

Ci sono casi nei quali un HA deve contattare necessariamente il

MSA, ad esempio quando avviene uno scambio IKEv2 (cfr. §2.3.2) che trasporta payload EAP che devono terminare su un server AAA oppure quando un HA voglia verificare se il profilo di un utente, gestito e conservato presso l'MSA, consenta l'autorizzazione al servizio MIPv6.

La definizione di un'interfaccia di comunicazione tra HA e MSA, ma anche tra MSP ed MSA quando queste entità sono separate e gestite da operatori diversi, non è affrontata in [9] e [10]. Il gruppo di lavoro MIP6 ha però prodotto un documento intitolato `AAA Goals for Mobile IPv6` [13] che illustra i requisiti per un'interfaccia con la piattaforma AAA. Questo documento è stato da utilizzato come riferimento nella fase di progettazione dell'interfaccia HA-AAA che presentiamo nel Capitolo 4 e che è l'argomento centrale di questo lavoro di tesi.

Capitolo 3
Il protocollo DIAMETER

3.1 Introduzione

Storicamente, i protocolli di Autenticazione, Autorizzazione ed Accounting (AAA) sono stati sviluppati per la necessità da parte degli ISP di disciplinare l'accesso di terminali dial-up nei primi anni di sviluppo di Internet. Il protocollo più affermato in tale ambito è senz'altro RADIUS (Remote Access Dial-In User Service) [14], sviluppato dall'IETF a partire dagli inizi degli anni '90 e divenuto lo standard attuale *de facto*.

Ma proprio perché nato per il dial-up, RADIUS non offre gli strumenti necessari per gestire applicazioni più sofisticate e servizi innovativi (come ad esempio Mobile IPv6) emersi negli ultimi anni.

Il documento [15] descrive i criteri per sviluppare protocolli AAA di nuova generazione. Tra le nuove funzionalità progettate ricordiamo:

- trasporto affidabile dei dati a livello di trasporto (livello 4) o applicazione (livello 7)
- possibilità di scoprire dinamicamente i nodi AAA e negoziazione

delle funzionalità;

- definizione di meccanismi di failover, ovvero funzionalità per far fronte al possibile malfunzionamento di un apparato dell'infrastruttura AAA;
- supporto per agenti di redirezione (relay) dei messaggi di AAA;
- possibilità per i server di inviare in maniera asincrona richieste ai client (ad esempio per richiedere lo stato delle sessioni degli utenti attivi);
- avanzato supporto per il roaming;

Diameter è il protocollo AAA di nuova generazione progettato in ambito IETF per rispondere ai requisiti suddetti e destinato a diventare il successore del protocollo RADIUS. Diameter è definito nella sua versione base nell'RFC 3588 [16] che descrive le funzionalità comuni a tutti i nodi Diameter. Specifiche funzionalità (accesso alla rete, autenticazione EAP, etc.) sono definite in documenti separati. Questo schema modulare facilita l'estensibilità del protocollo come, ad esempio, il supporto di nuove modalità di accesso che potranno presentarsi in futuro.

Dopo il rilascio nel 2003 delle prime specifiche Diameter (il già citato RFC 3588), è stato istituito in IETF il gruppo di lavoro DIME (DIameter Maintenance and Extensions) [17] per la manutenzione e l'estensione di Diameter. Inotre il gruppo di lavoro sta affrontando un lavoro di revisione del protocollo base [18] come spesso accade ai protocolli di nuova generazione prima che raggiungano una forma consolidata: la nuova versione aggiornata e rivista dell'RFC 3588 è prevista per agosto 2007.

Nel seguito del capitolo esamineremo le caratteristiche e le funzionalità principali di Diameter, approfondendo le parti più utili alla chiarificazione dei concetti espressi nei prossimi capitoli..

3.1.1 Terminologia e concetti base

Prima di procedere alla descrizione del protocollo Diameter spieghiamo ed esemplifichiamo il significato di alcuni termini basilari usati nel prosieguo.

La segnalazione Diameter avviene tra due **peer** (pari), così chiamati perché, al contrario della rigida gerarchia client/server di RADIUS, in Diameter è prevista nativamente la possibilità che anche un server possa inviare messaggi di richiesta,. I messaggi scambiati sono formati da un header seguito da una serie di Attribute Value Pairs (**AVP**), ovvero strutture dati contenenti un valore (Value) associato ad un parametro (Attribute). Gli attributi veicolano quindi le informazioni in un messaggio Diameter. Ogni messaggio è associato ad un **comando** specificato da un codice identificativo (*command code*) che ne definisce la natura (ad esempio una richiesta di autorizzazione).

Un'**applicazione** Diameter è un'estensione delle funzionalità del protocollo Diameter base. È necessario definire una nuova applicazione ogniqualvolta si debba definire un nuovo comando: ad esempio, per la gestione degli accessi in ambito dial-up o Terminal Access Server sono stati definiti i comandi AA-Request/Answer e di conseguenza è stata creata una nuova applicazione (NASREQ [19]).

Un **client** Diameter è tipicamente un apparato L2, L3 con

funzionalità di enforcemnt point (ad esempio un NAS). Un **server** Diameter è un apparato che gestisce le richieste AAA relative ad uno o più servizi (autenticazione EAP, Credit Control, etc.) e deve supportare necessariamente una o più applicazioni in aggiunta al protocollo base. Un server può essere di tipo *stateful* se mantiene e monitora lo stato delle sessioni attive o *stateless* in caso contrario.

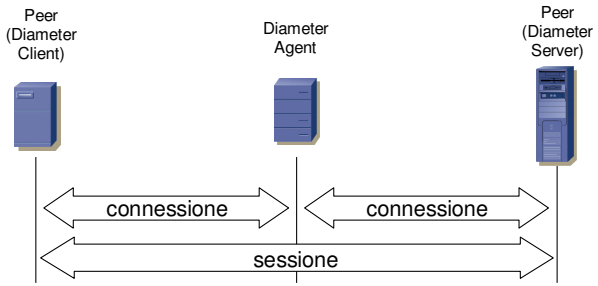

Figura 3.1 – Esempio di connessione e sessione Diameter

Oltre ai **nodi** Diameter appena citati esiste una terza tipologia di entità detta **agente** Diameter. Un agente è un apparato che svolge una funzione di tipo *relay*/*proxy*, che provvede ad inoltrare ad un server le richieste, o *redirect*, che restituisce al client informazioni sul server da contattare per una specifica richiesta. Una tipologia speciale di agent è quella di *translation* impiegata per convertire messaggi dal formato RADIUS a Diameter e viceversa. Il translation agent è stato definito per facilitare la migrazione di piattaforme AAA basate su RADIUS e salvaguardare gli investimenti degli operatori.

Una **connessione** è un collegamento al livello di trasporto tra due entità volto all'invio ed alla ricezione di messaggi Diameter. Una **sessione** Diameter è una sessione logica a livello applicativo che ha come estremi un client ed un server Diameter (cfr. Figura 3.1).

Il protocollo Diameter non definisce meccanismi di sicurezza propri ma stabilisce che due peer in comunicazione tra di loro utilizzino IPsec o in alternativa TLS per proteggere lo scambio dei messaggi. Per i dettagli sulla sicurezza in Diameter si rimanda alla sezione 13 di [16].

3.1.2 Diameter Base

Il protocollo Diameter base definisce i meccanismi basilari per lo scambio affidabile di messaggi tra due entità, i messaggi per instaurare ed abbattere una sessione e per la gestione degli errori.

Diameter utilizza come protocolli di trasporto TCP (Transmission Control Protocol) oppure SCTP (Stream Control Transmission Protocol), ovvero protocolli di livello 4 che garantiscono un trasporto affidabile dei dati in ambito best-effort.

Il protocollo Diameter base soddisfa i requisiti minimi per i protocolli AAA di nuova generazione, e in particolare esso fornisce nativamente:

- la negoziazione delle funzionalità tra entità;
- la notifica di eventi di errore, i quali si suddividono nei cosiddetti errori di protocollo (i.e. mancata consegna di una richiesta da parte di un relay agent) o di applicazione (errata composizione di un messaggio, contenuto di un AVP non corretto, etc.);

- l'estensibilità, grazie alla possibilità di sviluppare nuove applicazioni con nuovi comandi ed AVP in caso di necessità;
- le funzionalità di base comuni a tutte le applicazioni, tra cui la gestione delle sessioni utente e l'accounting.

La scoperta dei nodi (peers) è effettuata tramite configurazione manuale oppure con un meccanismo automatico. Quest'ultima avviene tramite il protocollo Service Location Protocol versione 2 (SLPv2 [20]) oppure con una richiesta DNS indicando il servizio "diameter" come record SRV.

Quando due entità Diameter instaurano una connessione al livello di trasporto, come prima operazione devono effettuare una *Capabilities Exchange*. Con questa segnalazione ogni nodo viene a conoscenza dell'identità del suo pari e delle funzionalità supportate: versione del protocollo Diameter in uso, applicazioni supportate, meccanismi di sicurezza, parametri di sessione e così via. I peer possono scambiarsi messaggi relativi ad applicazioni supportate da entrambi; nel caso in cui non ve ne fossero i peer possono scambiarsi solamente messaggi relativi al protocollo base, il cui supporto è obbligatorio per qualsiasi entità Diameter. Fanno eccezione i relay e redirect agent che devono supportare tutte le applicazioni standardizzate.

Ogni applicazione Diameter è identificata da un *Application ID* il cui valore è assegnato dallo IANA[1]. Ogni messaggio Diameter contiene nell'header (cfr. §3.1.3) l'Application ID che è anche utilizzato per il

[1] Lo IANA (Internet Assigned Numbers Authority) è l'organismo che supervisiona a livello mondiale l'allocazione degli indirizzi IP, i root server del servizio DNS e l'assegnazione di valori univoci per i protocolli di rete.

routing del messaggio (cfr. §3.1.5).

La comunicazione tra due entità Diameter è di tipo Request/Answer. Entrambe le entità possono inviare sia messaggi di tipo Request che messaggi di tipo Answer: ogni applicazione (anche quella base) stabilisce quale tipo di messaggio un'entità può inviare ed in quale situazione. Ad esempio, il messaggio AA-Request dell'applicazione NASREQ (relativo ad una richiesta di autenticazione/autorizzazione per l'accesso di utente) deve essere inviato dal client al server, il quale deve rispondere con un messaggio di AA-Answer. Invece il messaggio Abort-Session-Request del protocollo base, con il quale il server termina una sessione per un utente attivo (ad esempio a causa dello scadere della validità dell'autorizzazione), deve essere inviato dal server al client, il quale deve rispondere con un messaggio di Abort-Session-Answer.

Diameter prevede un meccanismo per rilevare l'interruzione della connessione a livello di trasporto tra due peer. Tale meccanismo prevede l'invio periodico di messaggi di *Device-Watchdog-Request* verso il peer durante i periodi di inattività (idle mode): se il peer non risponde prontamente con un messaggio di *Device-Watchdog-Answer,* Diameter deduce che la connessione non è più attiva. In tal caso, un nodo Diameter tenta di connettersi ad un altro peer per cercare di ripristinare il servizio. Questo meccanismo prende il nome di *failover.* Una volta connesso con un diverso nodo Diameter, le richieste pendenti al momento del failure vengono ritrasmesse tramite la nuova connessione per garantire la continuità del servizio di AAA. Periodicamente il nodo tenta comunque di ristabilire una connessione con il nodo disconnesso, ed in caso di successo

può ritornare a scambiare messaggi con tale nodo, operando un cosiddetto *failback*. Durante queste procedure potrebbero prodursi messaggi duplicati, ma il protocollo Diameter ha previsto strumenti per il riconoscimento e il corretto processing di tali messaggi, come riportiamo in §3.1.3.

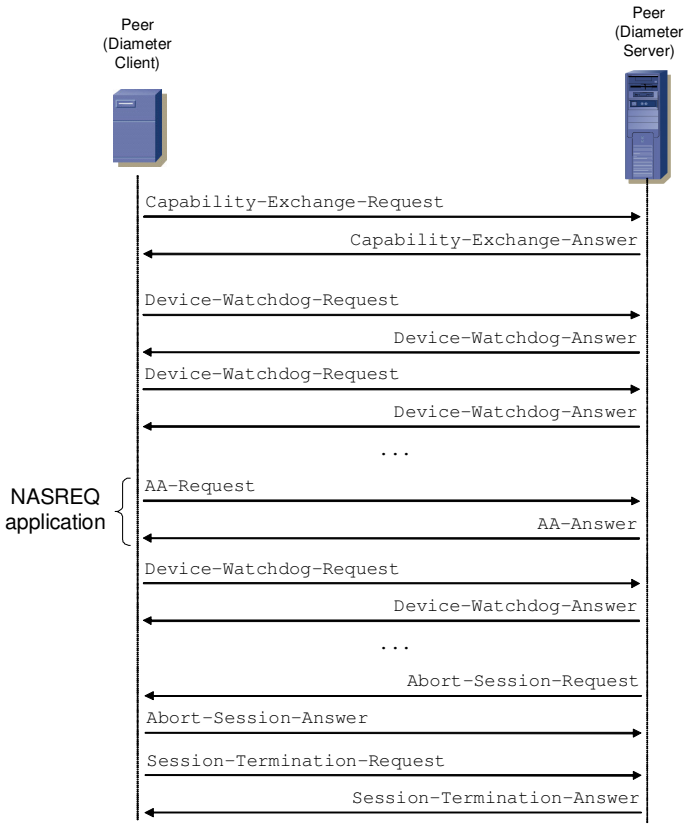

Figura 3.2 – Esempio di scambio di messaggi Diameter

In Figura 3.2 è illustrata una sessione Diameter di esempio. Come prima cosa, i due peer si scambiano i messaggi di Capabilities Exchange, successivamente sono trasmessi i messaggi di Watchdog per la verifica della connessione. Nel nostro esempio abbiamo poi raffigurato lo scambio di messaggi AA-Request/Answer dell'applicazione NASREQ per una richiesta di autenticazione/autorizzazione all'accesso di un utente. Terminato tale scambio, i due peer tornano a scambiarsi messaggi di Watchdog. Ad un certo momento il server decide di terminare il servizio attivato in precedenza, informando il client con un messaggio di Abort-Session-Request al quale segue un Abort-Session-Answer. Il client infine invia un messaggio di Session-Termination-Request per informare il server dell'effettiva fine del servizio, ricevendo in risposta una Session-Termination-Answer.

3.1.3 Header

Ogni messaggio Diameter è composto da un header di lunghezza fissa pari a 20 byte, il cui formato è mostrato in Figura 3.3.

Version è sempre posto ad 1 ad indicare la versione 1 del protocollo Diameter, quella attuale standardizzata dall'RFC 3588. *Message Length* indica la lunghezza in byte del messaggio Diameter comprensivo dell'header.

Successivamente sono presenti otto flag, quattro dei quali riservati per usi futuri:

- R = 1 indica un messaggio di tipo Request, altrimenti un messaggio di tipo Answer;

- P = 1 indica che il messaggio può non essere proxato o rediretto dal ricevente (ovvero può essere ad esempio re-diretto); se posto a 0 …
- E = 1 indica che il messaggio contiene un errore di tipo protocollare; non può essere settato in messaggi di tipo Request;
- T = 1 indica un messaggio ritrasmesso, quando ad esempio una richiesta non ha ricevuto risposta e viene inviata nuovamente.

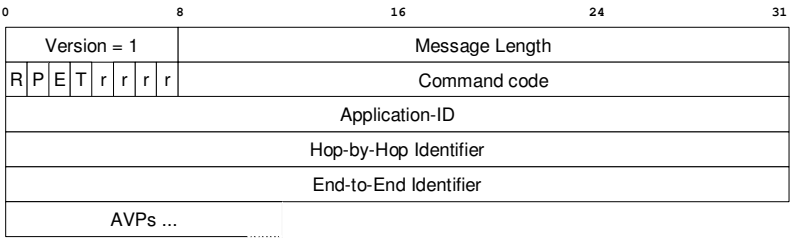

Figura 3.3 – Header di un messaggio Diameter

Command code è un campo di 3 ottetti ed indica il codice del comando associato al messaggio (e.g. request, answer).

Application-Id è un campo di 4 ottetti ed indica a quale applicazione è associato il messaggio. Gli Application ID ad oggi definiti sono:

- Diameter base 0
- applicazione NASREQ (RFC…) 1
- applicazione Mobile-IP (IPv4) 2
- accounting Diameter base 3
- applicazione Diameter Credit Control 4

- applicazione Diameter EAP 5
- applicazione Diameter SIP 6
- applicazione Relay 0xffffffff

Hop-by-hop Identifier è un campo di tipo intero senza segno su 32 bit. Il valore contenuto è settato dal mittente nei messaggi di Request per potervi associare le corrispondenti Answer: pertanto il peer che risponde con un Answer deve copiare il valore Hop-by-hop presente nel corrispondente messaggio di Request. Tipicamente in questo campo è presente un numero monotonicamente crescente a partire da un numero generato in modo casuale.

End-to-end Identifier è un campo di tipo intero senza segno su 32 bit ed è utilizzato per rilevare possibili messaggi duplicati. Anche questo valore è settato dai mittenti di messaggi Request e deve essere copiato nelle relative Answer. L'End-to-end Identifier non deve essere modificato dagli agenti Diameter.

Seguono poi gli AVP associati al particolare comando così come definiti dal protocollo Diameter base oppure dall'applicazione specifica.

3.1.4 Comandi

Ad ogni coppia di messaggi di tipo Request/Answer è associato un comando. Riportiamo di seguito i comandi definiti nell'RFC 3588:

Command-Name	Abbrev.	Code
Capabilities-Exchange-Request	CER	257
Capabilities-Exchange-Answer	CEA	257
Re-Auth-Request	RAR	258
Re-Auth-Answer	RAA	258
Accounting-Request	ACR	271
Accounting-Answer	ACA	271
Abort-Session-Request	ASR	274
Abort-Session-Answer	ASA	274
Session-Termination-Request	STR	275
Session-Termination-Answer	STA	275
Device-Watchdog-Request	DWR	280
Device-Watchdog-Answer	DWA	280
Disconnect-Peer-Request	DPR	282
Disconnect-Peer-Answer	DPA	282

Tabella 3.1 – Comandi Diameter Base

Si può notare come il protocollo base non definisca nessun comando specifico di tipo "applicativo" (ad esempio non è previsto alcun comando per autorizzare l'accesso).

La grammatica con cui sono definiti comandi del protocollo Diameter è la Augmented Backus-Naur Form (ABNF) [21], che illustriamo brevemente. I simboli di maggiore e minore indicano elementi obbligatori ed in posizione fissa all'interno del messaggio, ad esempio l'header. Gli elementi tra parentesi graffe sono obbligatori ma possono apparire in qualunque posizione nel messaggio, mentre quelli tra parentesi quadre sono opzionali. L'asterisco a fianco di un elemento significa che tale elemento può comparire un numero di volte compreso tra 0 ed infinito. L'ultimo elemento tra parentesi quadre (AVP) indica qualsiasi altro AVP oltre a

quelli elencati che non sia in conflitto con gli elementi obbligatori, in posizione fissa e non.

Di seguito descriviamo i comandi di Diameter base più utili alla comprensione del lavoro di tesi.

3.1.4.1. Session-Termination

Il comando Session-Termination è utilizzato per informare un server stateful della terminazione di una sessione relativa ad un utente.

```
<STR> ::= < Diameter Header: 275, REQ, PXY >
                < Session-Id >
                { Origin-Host }
                { Origin-Realm }
                { Destination-Realm }
                { Auth-Application-Id }
                { Termination-Cause }
                [ User-Name ]
                [ Destination-Host ]
              * [ Class ]
                [ Origin-State-Id ]
              * [ Proxy-Info ]
              * [ Route-Record ]
              * [ AVP ]
```

Qui sopra è presentato il Session-Termination-Request (STR), caratterizzato principalmente, oltre che dal Session-ID AVP che identifica la sessione da terminare, dall'AVP obbligatorio *Termination-Cause* contienente la motivazione per la terminazione del servizio (disconnessione dell'utente, fine della validità dell'autorizzazione per l'utente, etc.). È possibile inserire opzionalmente lo username dell'utente per la sessione in esame.

Di seguito mostriamo la forma del Session-Termination Answer (STA):

```
<STA>  ::= < Diameter Header: 275, PXY >
                  < Session-Id >
                  { Result-Code }
                  { Origin-Host }
                  { Origin-Realm }
                  [ User-Name ]
                * [ Class ]
                  [ Error-Message ]
                  [ Error-Reporting-Host ]
                * [ Failed-AVP ]
                  [ Origin-State-Id ]
                * [ Redirect-Host ]
                  [ Redirect-Host-Usage ]
                  [ Redirect-Max-Cache-Time ]
                * [ Proxy-Info ]
                * [ AVP ]
```

3.1.4.2. Abort-Session

Il comando Abort-Session è inviato da un server (stateful) ad un client per terminare prima del termine una sessione relativa ad un utente.

```
<ASR>  ::= < Diameter Header: 274, REQ, PXY >
                  < Session-Id >
                  { Origin-Host }
                  { Origin-Realm }
                  { Destination-Realm }
                  { Destination-Host }
                  { Auth-Application-Id }
                  [ User-Name ]
                  [ Origin-State-Id ]
                * [ Proxy-Info ]
                * [ Route-Record ]
                * [ AVP ]
```

Come si vede dalla grammatica del comando Abort-Session-Request (ASR), non è presente alcun AVP per motivare la terminazione della sessione utente. La sessione che il server vuole terminare è indicata nel Session-Id AVP. Opzionalmente, il server può inserire lo username dell'utente relativo alla sessione nello User-Name AVP.

Di seguito mostriamo la forma dell'Abort-Session-Answer (ASA):

```
<ASA>  ::= < Diameter Header: 274, PXY >
                 < Session-Id >
                 { Result-Code }
                 { Origin-Host }
                 { Origin-Realm }
                 [ User-Name ]
                 [ Origin-State-Id ]
                 [ Error-Message ]
                 [ Error-Reporting-Host ]
               * [ Failed-AVP ]
               * [ Redirect-Host ]
                 [ Redirect-Host-Usage ]
                 [ Redirect-Max-Cache-Time ]
               * [ Proxy-Info ]
               * [ AVP ]
```

3.1.5 Routing dei messaggi

I messaggi Diameter sono indirizzati verso le corrette destinazioni utilizzando il nome del dominio del destinatario e l'applicazione alla quale messaggio fa riferimento. La tabella di routing di un nodo Diameter è infatti denominata *Realm-Based Routing Table* (tabella di routing basata sui domini) della quale dettagliamo i campi più importanti:

- nome di dominio (tipicamente usato come chiave primaria per la ricerca in tabella);

- identificativo di applicazione (tipicamente usato come chiave secondaria per la ricerca in tabella);
- azione locale (indica come trattare i messaggi):
  1. LOCAL: i messaggi Diameter vengono elaborati localmente; è il caso di client e server.
  2. RELAY: i messaggi Diameter vengono inoltrati ad un server collegato ad un hop di distanza, senza modificare alcun AVP dedicato al routing.
  3. PROXY: i messaggi Diameter vengono inoltrati ad un server collegato ad un hop di distanza, ma l'entità locale può applicare regole locali includendo nuovi AVP nel messaggio prima di inoltrarlo.
  4. REDIRECT: i messaggi Diameter vengono restituiti al mittente con l'indicazione del nome del server da contattare per il messaggio in esame.
- Server Identifier (uno o più server verso i quali instradare il messaggio).

La tabella di routing può avere una voce di default da utilizzare nel caso in cui un messaggio sia processabile con alcuna disciplina di routing presente nella tabella.

Gli agenti Diameter possono avere sia una lista dei domini e delle applicazioni supportate localmente sia una lista dei domini e delle applicazioni raggiungibili esternamente. Quando viene ricevuta una richiesta relativa ad un dominio e/o ad un'applicazione non supportata localmente, il messaggio è re-indirizzato verso l'entità configurata nella

routing table.

Per maggiori dettagli si rimanda al documento [16].

3.1.6 Accounting base

Il protocollo di accounting descritto in [16] si basa su un modello orientato al server, con funzionalità per l'invio in tempo reale di informazioni di accounting. Con "modello orientato al server" si intende che i client che generano i dati di l'accounting interrogano un server (delle autorizzazioni oppure dell'accounting) per conoscere la modalità di consegna dei dati stessi.

Un nodo Diameter che ha ricevuto un messaggio di autenticazione e/o autorizzazione avvenuta con successo dal proprio Home AAA server deve raccogliere le informazioni di accounting per la relativa sessione. I dati sono inviati all'Home AAA server con un messaggio di Accounting-Request (AAC), il quale risponde con un messaggio di Accounting-Answer (AAA) con un opportuno Result-Code AVP a conferma dell'avvenuta ricezione.

Ogni elemento di accounting deve contenere il Session-Id AVP e, se disponibile, lo User-Name AVP. Sono possibili diversi tipi di elementi per l'accounting a seconda delle direttive sulla generazione e consegna dei dati stabilite dal server per l'accounting.

Se il servizio è di tipo "one time event", ovvero il servizio inizia e termina simultaneamente (ad es. una transazione on-line), allora l'Accounting-Record-Type AVP deve essere presente e posto al valore `EVENT_RECORD`. Se il servizio ha una durata diversa da zero, allora

l'Accounting-Record-Type deve contenere i valori `START_RECORD`, `STOP_RECORD` o `INTERIM_RECORD`.

Nei casi in cui siano client multipli ad essere coinvolti nel servizio (ad es. con Mobile IPv4/IPv6), ognuno di essi utilizza un Session ID diverso per il servizio relativo ad uno stesso utente e tale Session ID non è più utilizzabile come identificatore univoco per l'accounting. In questi casi il server comprende che i messaggi di Accounting Request sono relativi a sessioni già esistenti e include nelle risposte l'Acct-Multi-Session-Id AVP, il quale sarà utilizzato per correlare i dati di accounting.

Per le specifiche regole di accounting, definizione di sessione e/o multi-sessione il protocollo base rimanda alle singole applicazioni Diameter.

3.1.7 Attribute Value Pairs

Gli AVP trasportano sono i contenitori delle informazioni di autenticazione, autorizzazione, accounting, routing, sicurezza così come pure dei dati di configurazione.

Ogni AVP è naturalmente allineato ad una parola di 32 bit tranne per i tipi OctetString, per i quali è richiesto l'inserimento del minimo numero di zeri per l'allineamento a 32 bit; questo padding è trascurato nel valore del campo AVP Length.

Il campo *AVP Code*, combinato col campo opzionale Vendor-ID, identifica univocamente il tipo di attributo. I numeri da 1 a 255 sono riservati per la compatibilità con il protocollo RADIUS (il campo Vendor-ID deve essere lasciato vuoto). I numeri da 256 in avanti sono assegnati

dallo IANA e reperibili in Internet [22].

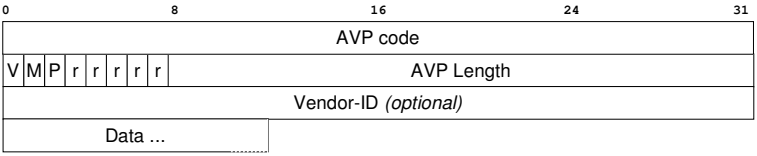

Figura 3.4 – Formato di un AVP

Degli otto *flag* presenti, tre sono assegnati ed i restanti cinque riservati per usi futuri (normalmente settati a 0):

- V (Vendor-Specific) = 1 indica che il campo opzionale Vendor-ID è presente; in questo caso il codice dell'AVP appartiene allo spazio di indirizzi specifico del Vendor proprietario del comando Diameter;
- M (Mandatory) = 1 indica che la presenza dell'AVP nel comando è obbligatoria;
- P (Protection) = 1 indica la presenza di un meccanismo di protezione per la sicurezza end-to-end.

Il campo *AVP Length* è su 3 ottetti ed indica la lunghezza in byte dell'intero AVP.

Il campo *Vendor-ID*, come detto, è presente solo se il flag 'V' è settato ad 1 e contiene il valore "SMI Network Management Private Enterprise Codes" assegnato dallo IANA al Vendor.

Il campo *Data* può essere composto da 0 o più byte contenenti informazioni specifiche dell'attributo. Il formato e la lunghezza di questo campo sono determinati dai precedenti campi AVP Code e AVP Length.

Tra i formati possibili per questo campo vi è, tra gli altri, il formato *Grouped*, che prevede come campo dati uno o più AVP in sequenza ognuno dei quali completamente strutturato (codice, lunghezza, flags, Vendor-ID se presente, dati). In questo caso il valore del campo AVP Length è dato da 8 (12 se il bit 'V' e settato ad 1) più la lunghezza totale in byte di tutti gli AVP inclusi nel campo Data.

Di seguito descriviamo gli AVP più importanti definiti dal protocollo base di Diameter (RFC 3588).

3.1.7.1. Session-Id

Il Session-Id, codice AVP 263, contiene dati di tipo UTF8String[2] ed è utilizzato per identificare una specifica sessione. Il valore del Session ID deve essere globalmente e temporalmente univoco al fine di identificare una sessione utente senza far riferimento a qualsiasi altra informazione (ad es. lo username), e può essere utilizzato per correlare dati per l'accounting con le relative informazioni di autenticazione.

Il Session ID è impostato dal nodo Diameter che inizia la sessione, ovvero nella maggior parte dei casi dal client. Il valore è dato dalla concatenazione dell'identità del mittente nella forma di DiameterURI (per il formato dell'identità cfr. §3.1.7.15), dei 32 bit più significativi seguiti dai 32 meno significativi di un valore monotonicamente crescente su 64 bit e di un valore opzionale specifico dell'implementazione. L'inizializzazione del valore su 64 bit potrebbe essere fatta in questo modo: all'avvio del

[2] La sigla sta per UCS/Unicode Translation Format su 8 bit, una codifica per caratteri a lunghezza variabile

client/server/agente Diameter può essere inserito nei 32 bit più significativi il tempo corrente e i 32 bit restanti possono essere settati a tutti 0. Questa inizializzazione può prevenire la sovrapposizione di Session ID, anche a seguito di reboot.

Per chiarezza riportiamo la struttura di un Session ID:

```
<DiameterIdentity>;<high 32 bits>;<low 32 bits>
[;<optional value>]
```

e due esempi, uno con ed uno senza il valore opzionale:

```
ap.acme.com;1876543211;528;mobile@200.1.1.89

ap.acme.com;1876543211;528
```

3.1.7.2. Origin-Host

L'Origin-Host, codice AVP 264, deve essere presente in ogni messaggio Diameter. Il contenuto di questo AVP rappresenta l'identità del creatore del messaggio, e non può essere modificato da agenti di tipo relay. Questo AVP dovrebbe essere posto quanto più possibile vicino all'header.

3.1.7.3. Origin-Realm

L'Origin-Realm, codice AVP 296, deve essere presente in ogni messaggio Diameter. Il campo dati individua il dominio del creatore del messaggio. Anche questo AVP dovrebbe essere posto quanto più possibile vicino all'header.

3.1.7.4. Destination-Realm

Il Destination-Realm, codice AVP 283, non può essere presente in

messaggi di tipo Answer. Esso contiene il nome del dominio verso il quale il messaggio va indirizzato. I client Diameter usano per questo AVP il dominio estratto dallo User-Name AVP (cfr. §3.1.7.10). I server che inviano messaggi di tipo Request usano il valore dell'Origin-Realm AVP ricevuto in precedenza dal client destinatario della richiesta (a meno che non sia noto a priori). Quando presente, il contenuto di questo AVP è utilizzato per prendere decisioni sul routing del messaggio. Anche questo AVP dovrebbe essere posto quanto più possibile vicino all'header.

3.1.7.5. Destination-Host

Il Destination-Host, codice AVP 293, deve essere presente in tutti i messaggi asincroni inviati da agenti Diameter, può essere presente in messaggi di tipo Request e non deve essere presente in messaggi di tipo Answer. Quando questo AVP non è presente, i messaggi di richiesta vengono inviati a qualunque server che supporta l'applicazione indicata nell'header all'interno del dominio indicato nel Destination-Realm AVP.

3.1.7.6. Auth-Application-Id

L'Auth-Application-Id, codice AVP 258, contiene dati di tipo intero senza segno su 32 bit ed è utilizzato per indicare il supporto per l'autenticazione ed autorizzazione di un'applicazione. Questo AVP deve essere presente in messaggi appartenenti ad applicazioni Diameter alle quali è stato assegnato un Application ID, ovvero in tutti i messaggi diversi da quelli definiti nel protocollo Diameter Base.

3.1.7.7. Auth-Request-Type

L'Auth-Request-Type, codice AVP 274, è incluso in messaggi di tipo Request specifici dell'applicazione per richiedere la sola autenticazione, la sola autorizzazione o autenticazione e autorizzazione in sequenza. I valori associati alle tre opzioni sono:

- AUTHENTICATE_ONLY 1
- AUTHORIZE_ONLY 2
- AUTHORIZE_AUTHENTICATE 3

3.1.7.8. Origin-State-Id

L'Origin-State-Id, codice AVP 278, è un valore di tipo intero senza segno su 32 bit monotonicamente crescente che viene incrementato da ogni volta che un'entità Diameter si riavvia senza perdita dello stato precedente (ad esempio in caso di reboot). Questo AVP, se presente, deve essere associato all'entità indicata nell'AVP Origin-Host: se, ad esempio, un agente di tipo proxy modifica l'AVP Origin-Host, deve essere modificato di conseguenza anche l'Origin-State-Id.

3.1.7.9. Result-Code

Il Result-Code, codice AVP 268, contiene un valore di tipo intero senza segno su 32 bit, è contenuto nei messaggi di tipo Answer ed indica l'esito della richiesta corrispondente. L'esito della richiesta, un numero a quattro cifre decimali, è codificato da IANA e le varie possibilità sono raggruppate in classi discriminate dalla prima cifra:

- Informational 1xxx

- Success 2xxx
- Protocol Errors 3xxx
- Transient Failures 4xxx
- Permanent Failures 5xxx

Codici di tipo `Informational` sono utilizzati per informare che la richiesta non può essere soddisfatta, ed al richiedente è richiesta un'ulteriore azione prima dell'accoglimento della richiesta.

La classe `Success` comprende codici per esprimere il completamento con successo della richiesta corrispondente: ad esempio, il codice 2001 `DIAMETER_SUCCESS` è presente in caso di esito positivo per una richiesta.

I codici della classe `Protocol Errors` sono relativi ad errori che possono essere risolti su base per-hop, ad esempio da agenti proxy tra client e server (ad es., mancato supporto di comandi al ricevente, server occupato, indicazione di redirezione). Questi e solo questi tipi di errore devono essere segnalati in presenza del bit E settato ad 1 nell'header del messaggio Diameter.

Errori di tipo `Transient Failures` sono relativi a richieste che non possono essere soddisfatte al momento della ricezione, ma che potrebbero essere soddisfatte in futuro: ad esempio autenticazione rifiutata oppure mancanza di memoria per salvare richieste di sessioni di accounting.

La classe `Permanent Failures` segnala richieste fallite che non devono essere ritentate, come ad esempio nei casi di autorizzazione rifiutata (codice 5003 `DIAMETER_AUTHORIZATION_REJECTED`).

3.1.7.10. User-Name

Lo User-Name, codice AVP 1, contiene una stringa di caratteri codificata nel formato UTF-8 che rappresenta lo username secondo il formato Network Access Identifier (NAI).

3.1.7.11. Authorization-Lifetime

L'Authorization-Lifetime, codice AVP 291, ha un campo dati di tipo intero senza segno su 32 bit che indica il periodo in secondi di validità del servizio erogato all'utente prima che l'utente medesimo debba essere ri-autenticato e/o ri-autorizzato. Il valore zero indica che il client deve immediatamente istruire una ri-autenticazione/autorizzazione, oppure deve tipicamente passare alla fase successiva per metodi di autenticazione a più round trip. L'assenza di questo AVP in un messaggio oppure la presenza di tutti i 32 bit del campo dati settati ad 1 implicano che il servizio non necessita di ri-autenticazione/autorizzazione. Se l'Authorization-Lifetime AVP compare in un messaggio assieme al Session-Timeout AVP (cfr. §3.1.7.13), il valore inserito nel secondo non può essere minore o uguale a quello nel primo.

Questo AVP può essere inserito dal client in messaggi di tipo Request come suggerimento del massimo valore di lifetime che è disposto ad accettare; resta inteso che il server può restituire un valore uguale oppure inferiore a quello suggerito.

3.1.7.12. Auth-Grace-Period

L'Auth-Grace-Period, codice AVP 276, ha un campo dato di tipo

intero senza segno su 32 bit che indica il numero di secondi che il server Diameter attende dopo la scadenza dell'Authorization Lifetime prima di liberare le risorse della relativa sessione.

3.1.7.13. Session-Timeout

Il Session-Timeout, codice AVP 27, ha un campo dato di tipo intero senza segno su 32 bit che indica la durata in secondi della sessione relativa al servizio fornito all'utente. Il valore zero oppure l'assenza di questo AVP indica che la relativa sessione ha durata illimitata.

Anche questo AVP può essere inserito dal client in messaggi di tipo Request come suggerimento del massimo valore per il Session Timeout che è disposto ad accettare; resta inteso che il server può restituire un valore uguale oppure inferiore a quello suggerito.

3.1.7.14. Auth-Session-State

L'Auth-Session-State, codice AVP 277, specifica se lo stato della sessione deve essere mantenuto o meno. Il client può includere questo AVP come suggerimento per il server, ma il valore restituito dal server nella risposta è vincolante. Le possibilità sono:

- STATE_MAINTAINED 0
- NO_STATE_MAINTAINED 1

3.1.7.15. Redirect-Host

Il Redirect-Host, codice AVP 292, contiene il nome del server, codificato nel formato DiameterURI, al quale re-indirizzare una richiesta. Il

formato del nome del server segue le regole di sintassi Uniform Resource Identifier (URI) specificate a pag. 44 di [16].

3.1.7.16. Accounting-Record-Type

L'Accounting-Record-Type, codice AVP 480, determina il tipo di dato di accounting contenuto nel messaggio:

- EVENT_RECORD 1
- START_RECORD 2
- INTERIM_RECORD 3
- STOP_RECORD 4

3.1.7.17. Acct-Interim-Interval

L'Acct-Interim-Interval, codice AVP 85, è di tipo intero senza segno su 32 bit ed è inviato dal Home server Diameter delle autorizzazioni al client. Le informazioni contenute nell'AVP servono al client per decidere come e quando produrre messaggi di accounting:

- se l'Acct-Interim-Interval non è presente oppure è presente ma con il campo dati posto a zero (0), allora il client deve produrre messaggi di accounting con Accounting-Record-Type di tipo EVENT_RECORD, START_RECORD e STOP_RECORD.
- se l'Acct-Interim-Interval è presente con un valore del campo dati diverso da zero, allora il client deve inviare dei messaggi di accounting periodici (INTERIM_RECORD) tra l'inizio e la fine della fase di accounting ad intervalli di tempo pari al valore indicato nel campo dati.

3.1.7.18. Acct-Multi-Session-Id

L'Acct-Multi-Session-Id, codice AVP 50, è di tipo UTF8String (cfr. Session-ID §3.1.7.1) ed è utilizzato per correlare sessioni multiple di accounting con la corretta sessione di autenticazione/autorizzazione.

Capitolo 4

Interazione tra Mobile IPv6 e la piattaforma AAA dell'operatore

4.1 Introduzione

Lo scopo di questo lavoro di tesi è la definizione di un'interfaccia HA – server AAA necessaria agli operatori per il controllo e la gestione del servizio Mobile IPv6. Quest'interfaccia è utilizzata, ad esempio, da un HA per interrogare l'infrastruttura AAA al fine di ottenere le opportune autorizzazioni prima di erogare il servizio MIPv6 ad un utente. Il protocollo utilizzato per tale interfaccia è Diameter, il protocollo AAA di nuova generazione illustrato nel capitolo precedente.

Il punto di partenza è stato il lavoro del gruppo DIME di IETF che ha formalizzato in [13] i requisiti dell'interfaccia AAA per Mobile IPv6 e ha abbozzato in [23] una soluzione tecnica per tale interfaccia.

4.2 L'interfaccia HA – server AAA

All'interno del DIME le proposte più significative per la scelta dell'applicazione Diameter da utilizzare nell'interfaccia HA – server AAA sono state:

1) il reimpiego dell'applicazione Diameter EAP [24] con opportune modifiche, ovvero con l'introduzione di specifici comandi ed AVP per segnalare autenticazioni e autorizzazioni relative alla mobilità;;
2) lo sviluppo di una nuova applicazione basata su Diameter EAP ma ma svincolata da essa per evitare di dover aggiornare l'applicazione "madre";
3) la definizione di un'applicazione ex-novo.

Dopo un lungo dibattito sulla mailing list del DIME, è stato scelto un compromesso tra gli approcci 1) e 3) separando la fase di autenticazione da quella di autorizzazione e accounting. Per l'autenticazione è stato proposto l'utilizzo dell'applicazione Diameter EAP mentre per l'autorizzazione e l'accounting è stato deciso di definire una nuova applicazione Diameter. Le prinicipali motivazioni che hanno portato a questa scelta sono state:

- l'impossibilità di estendere l'applicazione Diameter EAP con comandi ed AVP specifici per MIPv6, in quanto ciò avrebbe comportato necessariamente la definizione di una nuova applicazione (cfr. §3.1.1);
- la possibilità di definire un'unica interfaccia HA – server AAA per gli scenari split ed integrated;
- la modularità dell'approccio garantisce la possibilità di utilizzare

altre forme di autenticazione (ad esempio IKE-PSK), fermo restando l'uso suggerito di EAP.

Il nuovo documento prodotto dal DIME nell'ottobre 2006 [25] formalizza questo approccio che separa l'autenticazione dall'autorizzazione e accounting e delinea un'architettura per l'interazione HA – server AAA:

- per l'autenticazione è suggerito l'uso dell'applicazione Diameter EAP, alla quale non vengono apportate modifiche a meno dell'utilizzo del valore 1 (AUTHENTICATE_ONLY) nell'AVP Auth-Request-Type dei messaggi Diameter-EAP-Request/Answer (DER/DEA). Questa informazione può essere utilizzata dal server AAA che processa la richiesta di autenticazione come hint che la richiesta è relativa ad un utente del servizio Mobile IPv6. Il server AAA, infatti, potrebbe non essere un server dedicato esclusivamente a gestire utenti MIPv6.
- per l'autorizzazione e l'accounting è definita una nuova applicazione Diameter denominata "Mobile IPv6 Authorization Application".

Nel documento [25] restano aperte alcune questioni basilari, quali il timing per avviare la fase di autorizzazione/accounting relativamente alla fase autenticazione, il supporto per Authentication Protocol, la necessità di un cosiddetto "authentication token" per legare crittograficamente le fasi di autenticazione e autorizzazione (cfr. §4.6), la co-locazione dei server AAA coinvolti.

Il nostro lavoro cerca di colmare tali lacune proponedo soluzioni specifiche a queste issue ed inoltre introduce alcune estensioni per gestire alcune funzionalità avanzate.

4.3 Panoramica sulla nuova applicazione MIP6 Authorization

L'applicazione descritta in [25] definisce lo scambio di messaggi di autorizzazione Request/Answer tra HA e server Diameter. Il comando associato a tali messaggi è denominato "MIP6-Authorization". Il messaggio di richiesta, `MIP6-Authorization-Request (MAR)`, è inviato dal client (HA) al server AAA e la risposta, `MIP6-Authorization-Answer (MAA)`, è inviata indietro dal server al client. La caratteristica di questi messaggi è il valore dell'Auth-Request-Type AVP posto a 2 (`AUTHORIZE_ONLY`).

Nella nostra soluzione la fase di autorizzazione (scambio dei messaggi MAR/MAA) inizia una volta terminata con successo la procedura di autenticazione (ad esempio attraverso EAP). Più esattamente, (cfr. Figura 4.1) l'inivio del MAR è triggerato dall'arrivo della prima Binding Update inviata dell'utente come evento che provoca l'inizio dell'autorizzazione (per maggiori dettagli si veda §4.5). Come detto, l'Home Agent, che agisce da client Diameter, contatta il server del MSA (di seguito chiamato AAAH-MIP6 come in [25]) inviando un messaggio MAR contenente l'identità (nella forma di NAI) dell'utente da autorizzare. La risposta del server è contenuta nel messaggio MAA e in caso di successo il server AAAH-MIP6 comunica all'Home Agent in tale messaggio il periodo di validità dell'autorizzazione.

È da notare il fatto che le richieste di autorizzazione al servizio Mobile IPv6 potrebbero raggiungere un server AAA diverso da quello interrogato per l'autenticazione; il motivo risiede nel meccanismo di routing

dei messaggi Diameter che prende in considerazione anche l'Application-Id (cfr. §3.1.5). Per considerazioni su possibili problemi di sicurezza dovuti all'uso di due applicazioni diverse per autenticazione ed autorizzazione si rimanda al §4.6.

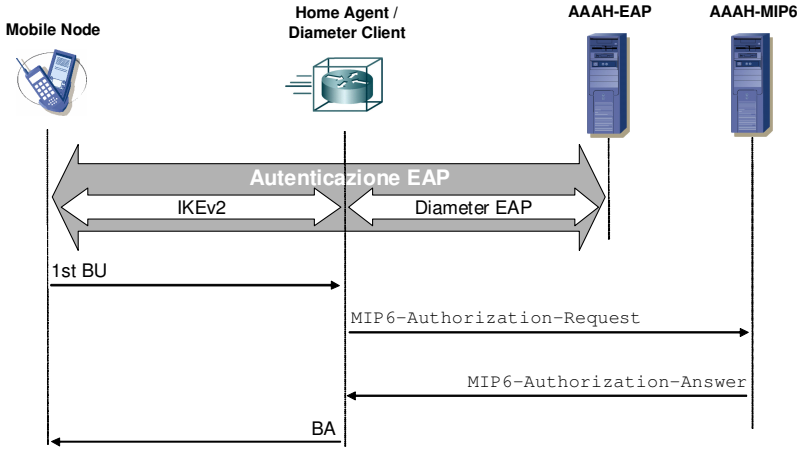

Figura 4.1 – Esempio di messaggi Diameter MIP6 Authorization

Terminata con successo la procedura di autorizzazione, il servizio Mobile IPv6 viene erogato al nodo mobile dall'HA, il quale inizia anche le corrispondenti operazioni di accounting che prevedono l'invio periodico o di tipo "one-time event " (cfr. §3.1.6) al server AAAH-MIP6 di messaggi di tipo Accounting-Record contenenti AVP specifici per il servizio Mobile IPv6.

4.4 Autenticazione

L'approccio modulare scelto per l'interfaccia HA – server AAA che separa le fasi di autenticazione e di autorizzazione/accounting permette, almeno in via teorica, l'utilizzo di diversi metodi di autenticazione. I metodi di autenticazione che prenderemo in considerazione per l'interfaccia HA-server AAA sono EAP, IKEv2 con Pre-Shared Keys (PSK) ed Authentication Protocol. Con l'esclusione dell'uso di EAP, che è descritta nel documento [25], l'autenticazione tramite IKE-PSK e il supporto per l'Authentication Protocol sono specifici del lavoro di questa tesi.

. Nel caso in cui l'MSA e l'ASA coincidano (integrated scenario) e che l'autenticazione per l'accesso alla rete sia basata su EAP è possibile derivare il materiale cifrato utilizzato per l'IKE-PSK e l'Authentication Protocol dallo scambio EAP effettuato per l'autenticazione all'accesso

4.4.1 Autenticazione con EAP over IKE

L'autenticazione basata su EAP (Extensible Authentication Protocol) è descritta in dettaglio in [25].

MN ed HA iniziano la conversazione (passi 1,2 in Figura 4.2.) con uno scambio di messaggi IKE_SA_INIT: in questa fase sono negoziati gli algoritmi di cifratura con lo scambio dei nonces e dei parametri per la crittografia di Diffie-Hellman. La fase IKE_AUTH (che inizia al passo 3) è volta all'autenticazione dei messaggi precedenti, con i quali MN e HA si scambiano le rispettive identità; al termine della fase IKE_AUTH verrà stabilita la prima CHILD_SA (passo 4) tra MN ed HA.

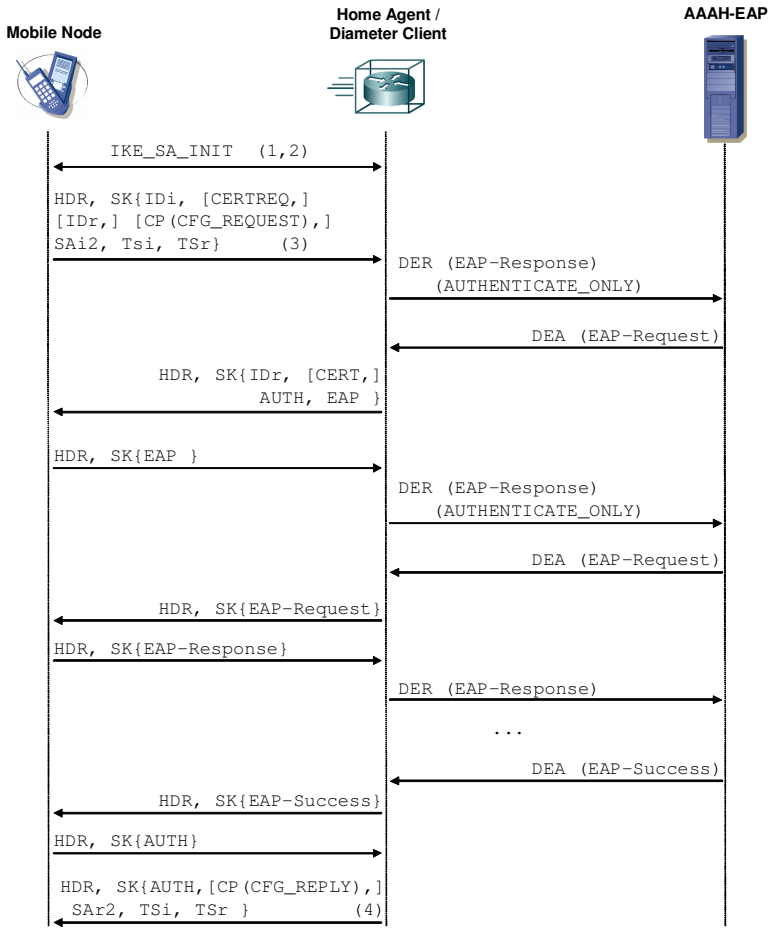

Figura 4.2 – Esempio di scambio di messaggi per autenticazione EAP

L'identità dell'utente/MN è inserita nel campo IDi del messaggio 3 (tipicamente nella forma di NAI), e la volontà del MN di essere autenticato

con un metodo EAP è indicata dall'omissione del campo AUTH nel medesimo messaggio Tipicamente, il MN autentica l'HA attraverso un certificato che viene richiesto includendo il campo [CERTREQ] nel messaggio 3. Durante il processo di autenticazione il MN può manifestare la volontà di ricevere un Home Address o un prefisso per la home network oppure può suggerirne uno, utilizzando il campo CFG_REQUEST nel messaggio 3.

L'Home Agent estrae il campo IDi dal messaggio 3 ed invia un messaggio Diameter-EAP-Request al server AAAH-EAP attraverso il nome del dominio presente nel NAI dell'utente: l'HA si comporta così da autenticatore in modalità pass-through, rispettando così la richiesta al punto 4.1 del documento dei goals per le interfacce con piattaforme AAA [13]. Nel messaggio DER sono presenti, tra gli altri, lo User-Name AVP contenente l'identità ricopiata dal campo IDi e l'Auth-Request-Type AVP contenente il valore 1 (AUTHENTICATE_ONLY). Il messaggio DER raggiunge il server AAAH-EAP relativo al dominio dell'utente/MN attraverso il routing AAA: il server AAAH-EAP processa le informazioni ricevute e risponde con un messaggio Diameter-EAP-Answer.

Al termine della fase di autenticazione EAP che, a seconda del metodo EAP utilizzato, può prevedere un numero variabile di scambi EAP-Request/Response, il server AAAH-EAP indica il risultato dell'autenticazione nel Result-Code AVP dell'ultimo messaggio DEA nel quale include anche il relativo messaggio EAP (EAP-Success oppure EAP-Failure). Successivamente, l'HA comunica al MN nell'ultimo messaggio IKEv2 (passo 4) l'Home Address o il prefisso della home network. Infine, è effettuato uno scambio di messaggi CREATE_CHILD_SA per il setup delle

IPsec Security Association (SA) per la successiva segnalazione relativa al protocollo Mobile IPv6 (invio di Binding Update, Binding Acknowledgement).

4.4.2 Autenticazione con IKE-PSK

IKEv2 (Internet Key Exchange version 2) [6] prevede una modalità PSK per l'autenticazione. L'autenticazione dell'initiator (MN del nostro caso) con IKE-PSK è possibile qualora entrambi i dispositivi siano in possesso dello stesso segreto condiviso. Sorge così il problema della distribuzione del materiale crittografico dato che, verosimilmente, le chiavi non possono essere distribuite in maniera statica poichè non è plausibile fornire le chiavi di tutti gli utenti a tutti gli i HA (che ricordiamo essere assegnati dinamicamente in fase di bootstrap). A complicare la situazione, c'è anche il fatto che l'HA assegnato al MN può, in generale, appartenere ad un dominino amministrativo diverso dall'Home MSP dell'utente.

Pertanto la nostra soluzione per il supporto della modalità di autenticazione con PSK prevede il recupero del materiale crittografico da parte dell'HA presso il server delle autenticazioni della mobilità (supposto co-locato con il server di accesso e di seguito denominato MSA-AAA). Per questo meccanismo abbiamo previsto l'uso dell'applicazione Diameter NASREQ [19] tra HA e server AAA.

Lo scambio di messaggi Diameter inizia successivamente alla ricezione del primo messaggio di tipo IKE_AUTH da parte dell'HA (step 4 di Figura 4.3) nel quale il MN ha inserito la propria identità per l'autenticazione nel campo IDi. L'HA invia un messaggio AA-Request (step 5) al server nel quale inserisce all'interno di uno User-Name AVP

l'identità ricevuta. Il valore dell'Auth-Request-Type AVP è posto ad 1 (AUTHENTICATE_ONLY) in quanto la segnalazione Diameter è volta solo ad autenticare il MN per il servizio Mobile IPv6, non ad autorizzarlo.

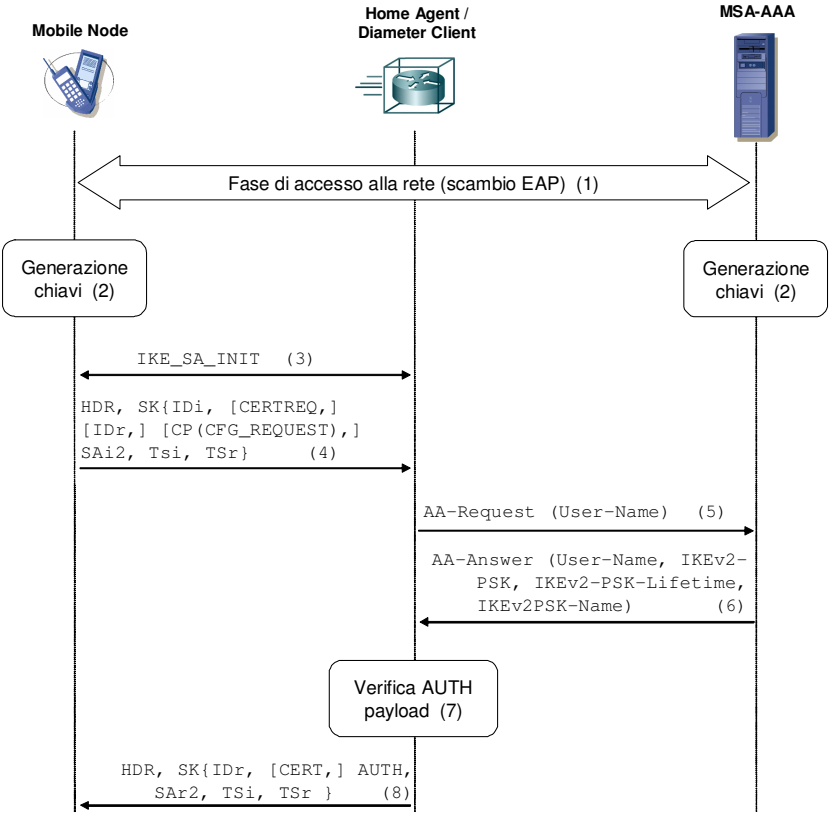

Figura 4.3 – Esempio di scambio di messaggi per autenticazione PSK

Il server AAA riceve la richiesta e sulla base dell'identità ricevuta

recupera il materiale PSK corrispondente; quindi invia la risposta AA-Answer (step 6) contenente la chiave richiesta, il lifetime corrispondente e il nome della chiave in appositi AVP da noi definiti (rispettivamente IKEx2-PSK AVP, IKEv2-PSK-Lifetime AVP e IKEv2-PSK-Name AVP). Inoltre nella risposta il server inserisce nello User-Name AVP l'identità che l'HA dovrà utilizzare nella successiva richiesta di autorizzazione da effettuarsi con un messaggio MIP6-Authorization-Request (per l'applicazione MIP6 Authorization si rimanda al §4.5).

Con la chiave recuperata, l'HA può così verificare l'AUTH payload (step 7) e generare la risposta per il MN terminando lo scambio di messaggi IKE_AUTH (step 8).

4.4.3 Supporto per Authentication Protocol

L'RFC 4285 [26] descrive un metodo per autenticare la segnalazione tra MN ed HA (messaggi BU/BA) evitando la complessa instaurazione di Security Association IPsec tra Mobile Node e Home Agent (cfr. Appendice A §A.6). I messaggi BU/BA vengono autenticati mediante una nuova `Mobility Message Authentication Option` (per i dettagli sull'opzione cfr §B.6 in Appendice B) che può essere di due tipi: MN-HA per l'autenticazione eseguita da MN e HA e MN-AAA per l'autenticazione eseguita tra MN e server AAA. Il contenuto delle opzioni è generato utilizzando una `shared-key-based mobility security association`, ovvero un segreto condiviso tra MN e HA per l'opzione MN-HA e tra MN e server AAA per l'opzione MN-AAA.

Similmente a quanto detto per l'autenticazione PSK, l'HA non

tipicamente ha alcun segreto condiviso con il MN. Pertanto la nostra soluzione prevede che l'HA recuperi dinamicamente il materiale crittografico necessario alla generazione e verifica dei dati di autenticazione contenuti nelle opzioni MN-HA presso il server AAA delle autenticazioni (MSA-AAA).

A differenza del caso precedente IKE-PSK, in questo caso è opportuno utilizzare la stessa applicazione Diameter MIP6 Authorization per la segnalazione tra HA e server AAA dato che con l'Authentication protocol l'utente è implicitamente autenticato tramite l'autenticazione dei messaggi di BA. La spiegazione dettagliata della soluzione è riportata in §4.5.2.2 nella sezione relativa all'autorizzazione.

4.4.4 Nuovi AVP definiti

Per il supporto delle funzionalità descritte nei paragrafi precedenti abbiamo definito i seguenti nuovi AVP:

- IKEv2-PSK AVP: contiene la chiave IKEv2-PSK;
- IKEv2-PSK-Lifetime AVP: contine il valore relativo alla validità temporale (lifetime) della chiave IKEv2-PSK;
- IKEv2-PSK-Name AVP: contiene il nome della chiave IKEv2-PSK;
- Binding-Update-Message AVP: contiene la BU inviata dal MN all'HA da autenticare con Authentication Protocol;
- MN-HA-Key AVP: contiene la chiave MN-HA;
- MN-HA-Key-Lifetime AVP: contiene il valore di lifetime della chiave MN-HA;
- MN-HA-Key-Name AVP: contiene il nome della chiave MN-HA.

4.5 Autorizzazione

Come già spiegato in precedenza, l'applicazione utilizzata per l'autorizzazione di un utente (MN) e la gestione delle relative sessioni è la Diameter MIP6 Authorization. Tipicamente la fase di autorizzazione segue una autenticazione conclusa con successo, in [25], però, non è chiarito come le due fasi di autenticazione e di autorizzazione debbano interagire. Nella nostra soluzione si è scelto di separare nettamente le due fasi, utilizzando come innesco della fase di autorizzazione l'arrivo all'Home Agent della prima BU inviata da un Mobile Node e quindi svincolandosi completamente da eventi di autenticazione. Questa scelta permette di colocare su uno stesso apparato più servizi che richiedano lo stesso tipo di autenticazione (e.g. EAP over IKE). Ad esempio se l'HA possiede anche funzionalità di VPN concentrator, l'utente potrebbe autenticarsi per instaurare una VPN e non per usufruire del servizio Mobile IPv6. Avendo completamente svincolato l'autenticazione da servizio associato, ciò è possibile senza ulteriori complicazioni.

Questa soluzione, però, non lascia al server MSA (di seguito denominato AAAH-MIP6 come in [25]) a possibilità di indicare esplicitamente al HA se il MN è autorizzato ad autoconfigurare il proprio Home Address o meno. Tale possibilità è indicata nel punto G5.2 del documento sui goals [13], requisito che suggerisce il completamento della la fase di autorizzazione al servizio MIPv6 prima che qualsiasi messaggio Mobile IPv6 sia scambiato (ad esempio una BU). Ma un MN può mandare una BU se e solo se esso ha stabilito una IPsec SA con l'HA, e per farlo deve conoscere parametri tra i quali il proprio l'Home Address.

4.5.1 Dettaglio dei messaggi

4.5.1.1. MIP6 Authorization Request (MAR)

Descriviamo il messaggio MAR con la grammatica ABNF (una breve descrizione della ABNF è presente nel §3.1.4) evidenziando in corsivo gli AVP da noi introdotti e spiegati nel seguito:

```
<MIP6-Authorization-Request> ::
             = < Diameter Header: TBD, REQ, PXY >
               < Session-Id >
               { Auth-Application-Id }
               { Origin-Host }
               { Origin-Realm }
               { Destination-Realm }
               { Auth-Request-Type }
               { User-Name }
               [ Destination-Host ]
               [ Origin-State-Id ]
               [ Authorization-Lifetime ]
               [ Auth-Grace-Period ]
               [ Session-Timeout ]
               [ Auth-Session-State ]
               [ MN-HomeAddress ]
               [ MN-CareofAddress ]
             * [ AVP ]
```

Il messaggio MAR è inviato da un client Diameter ad un server: ha il command code posto a TBD (to be defined), il bit R = 1 in quanto il messaggio è di tipo Request e il bit P = 1 per permettere al messaggio di essere inoltrato e/o ritrasmesso da agenti Diameter.

Da notare che l'Auth-Request-Type dovrà essere posto al valore decimale 2 corrispondente ad AUTHORIZE_ONLY, e che lo User-Name AVP è obbligatorio e contiene l'identità dell'utente da autorizzare.

Tra gli AVP opzionali si hanno Authorization-Lifetime, Auth-Grace-Period, Session-Timeout e Auth-Session-State che se inseriti dal client sono utilizzati come indicazioni dal server.

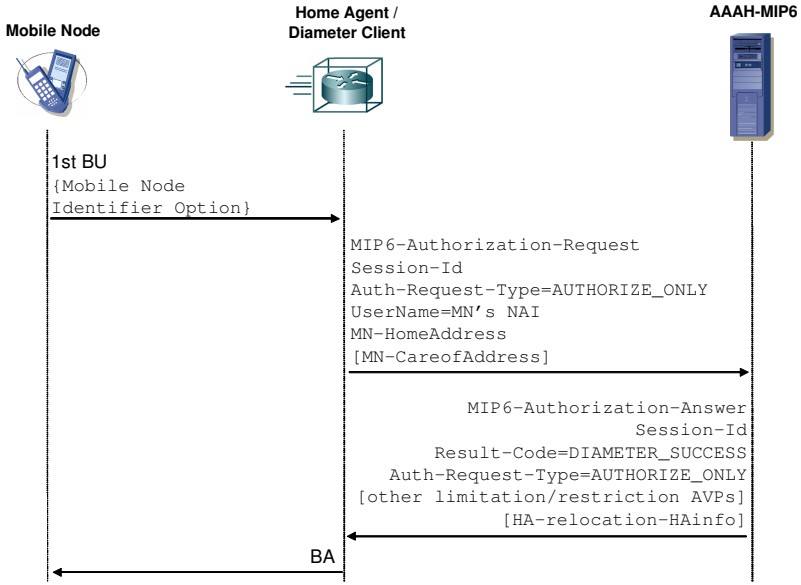

Figura 4.4 – Esempio di messaggi Diameter MIP6 Authorization

Nella nostra soluzione sono definiti, inoltre, due nuovi AVP opzionali:

- MN-HomeAddress AVP: contiene l'Home Address del MN. Tale informazione può essere inviata dal client al server AAAH affinché quest'ultimo possa aggiornare nel DNS l'FQDN del MN. Il fatto che sia l'AAAH ad eseguire l'aggiornamento DNS rende la soluzione

adatta anche per gli scenari nei quali non esiste una relazione fidata tra HA e il server DNS che gestisce il dominio del MN, ad esempio quando HA e MN fanno parte di domini amministrativi diversi.

- MN-CareofAddress AVP: contiene il Care-of Address primario del Mobile Node. L'informazione del CoA può essere utilizzata dal'AAAH, ad esempio, per implementare un sistema di HA reliability, ovvero un sistema che garantisca la continuità del servizio in caso di malfunzionamento dell'HA.

4.5.1.2. MIP6 Authorization Answer (MAA)

Descriviamo il messaggio MAA:

```
<MIP6-Authorization-Answer> ::
             = < Diameter Header: TBD, PXY >
               < Session-Id >
               { Auth-Application-Id }
               { Result-Code }
               { Origin-Host }
               { Origin-Realm }
               { Auth-Request-Type }
               [ User-Name ]
               [ Destination-Host ]
               [ Origin-State-Id ]
               [ Authorization-Lifetime ]
               [ Auth-Grace-Period ]
               [ Session-Timeout ]
               [ Auth-Session-State ]
               [ HA-relocation-HAinfo ]
             * [ AVP ]
```

Il messaggio MAA è inviato dal server AAAH al clienti in risposta ad un messaggio MAR: ha un command code pari a TBD e il bit P = 1

nell'header. Il Session-Id è identico a quello della Request corrispondente e il campo Result-Code contiene l'esito codificato della richiesta ricevuta. Auth-Request-Type è anch'esso pari a 2 (AUTHORIZE_ONLY) e negli AVP di autorizzazione (Authorization-Lifetime, Auth-Grace-Period, Session-Timeout) sono inviati i dati sulla validità dell'autorizzazione al servizio MIPv6 per l'utente.

4.5.2 Funzionalità aggiuntive della nostra soluzione

4.5.2.1. Ri-autorizzazione

Nella nostra soluzione si è definita una procedura, non determinata in Diameter base ma tipica delle applicazioni di autenticazione/ autorizzazione quali NASREQ, per ri-autorizzazione (periodicamente o in base a determinati eventi) l'utente da parte del client Diameter (HA). Questo requisito è presente nel goal G2.5 presente in [13]. L'esempio più tipico che determina una ri-autenticazione iniziata dal client è la scadenza del lifetime di autorizzazione concesso ad un utente.

La procedura studiata per la ri-autorizzazione iniziata dal client è mostrata in Figura 4.5: una volta avvenuto l'evento che fa scattare la ri-autorizzazione (scadenza del lifetime, nuovo CoA ricevuto, etc.), l'HA invia un messaggio MAR simile a quello per l'autorizzazione, con il Session-Id relativo alla sessione attiva dell'utente. Per evitare ambiguità il client può inserire anche lo User-Name AVP contenente l'identità dell'utente da ri-autorizzare. Il server AAAH-MIP6 processa la richiesta e in caso di successo risponde con un messaggio MAA del tutto simile a quello prodotto a seguito della prima autorizzazione.

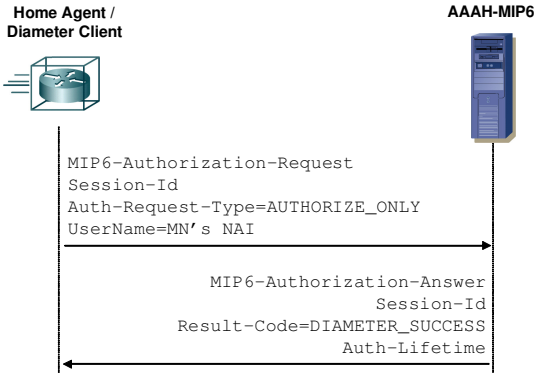

Figura 4.5 – Scambio di messaggi per la ri-autorizzazione

4.5.2.2. Supporto per RFC 4285

Dettagliamo di seguito la nostra soluzione per il supporto di Authentication Protocol. Il meccanismo prevede il recupero della chiave MN-HA presso il server MSA da parte dell'HA.

Per autenticarsi utilizzando Authentication Protocol MN ed HA devono autenticare i messaggi BU/BA attraverso la Mobility Message Authentication Option. Il MN non può però inserire la MN-HA Mobility Message Authentication Option nella prima BU, perché l'HA non può verificare i dati di autenticazione dell'opzione MN-HA non conoscendo (cfr. §4.4.3) ancora la chiave MN-HA.

Allora il MN inserisce nella prima BU la MN-AAA Mobility Message Authentication Option, opzione che l'HA inoltra al server MSA assieme all'identità del MN. In questo modo il server AAA deriva la chiave MN-HA relativa al MN, autentica il MN e restituisce all'HA la chiave MN-HA da utilizzare per i successivi scambi BU/BA col MN e la validità

temporale della chiave stessa.

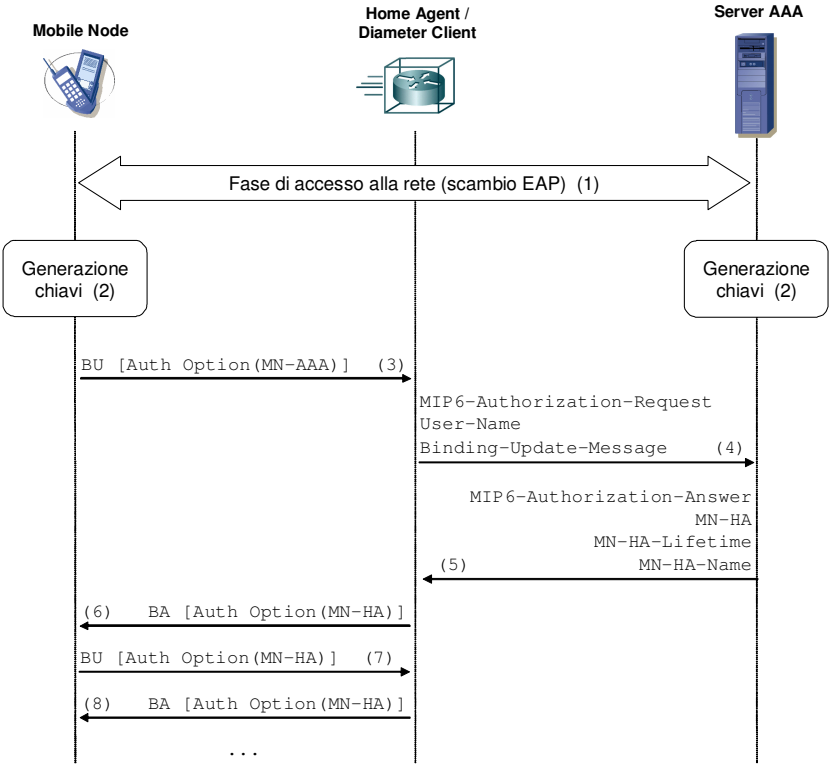

Figura 4.6 – Esempio di scambio di messaggi per autenticazione con RFC 4285

In Figura 4.6, al passo 3 il MN invia un messaggio di Binding Update all'HA inserendo una Mobility Message Authentication Option di tipo MN-AAA. Per l'autorizzazione e il recupero del materiale cifrato l'HA invia un messaggio MAR al server AAA contenente l'identità dell'utente

nello User-Name AVP e la BU inviata dal MN, inserita nel Binding-Update-Message AVP.

Tramite la chiave MN-AAA (derivata al passo 2) il server AAA autentica la BU e in caso di successo risponde (passo 4) con un messaggio MAA contenente la chiave MN-HA, la validità temporale ed il nome della stessa rispettivamente negli AVP MN-HA, MN-HA-Lifetime e MN-HA-Name. Attraverso la chiave MN-HA ricevuta, l'HA è in grado di autenticare i successivi messaggi scambiati con il MN fino allo scadere del lifetime senza contattare ulteriormente il server AAA. Terminato il periodo di validità della chiave il MN dovrà ripetere la procedura iniziale con l'invio di una BU autenticata con la chiave MN-AAA per permettere all'HA di recuperare una nuova chiave MN-HA.

4.5.2.3. Supporto per l'HA relocation

Con HA relocation si intende un insieme di procedure atte a permettere ad un nodo mobile di cambiare dinamicamente (cioè senza perdere le sessioni attive) l'HA utilizzato per il servizio Mobile IPv6. Il cambio di HA è gestito dall'MSP che determina quando una relocation debba avere luogo e segnala questa decisione attraverso l'infrastruttura AAA all'HA.

Una HA relocation può essere dovuto all'esigenza di ripartire il volume di traffico dovuto ai MN attivi su più HA dell'MSP, alla volontà di migliorare le prestazioni per il MN con l'utilizzo di un HA topologicamente più vicino al MN stesso, alla necessità di arrestare un HA per operazioni di manutenzione.

L'HA relocation può avvenire in qualsiasi momento successivo

all'inizio del servizio Mobile IPv6. In particolare, nello split scenario essa può verificarsi anche immediatamente dopo la fase di autorizzazione. Questo perché in uno scenario integrated l'HA viene assegnato al MN (e non scoperto da quest'ultimo – cfr. §2.3.1.1) ed è possibile allocare l'HA ottimale già in questa fase.

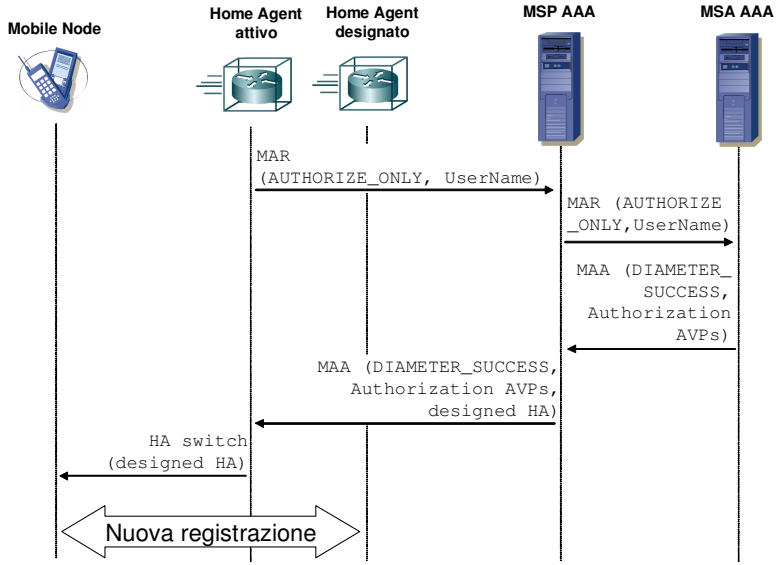

Figura 4.7 – Scambio di messaggi per HA relocation

Il procedimento attraverso il quale si articola una HA relocation è rappresentato in Figura 4.7: l'MSP verifica periodicamente, ad esempio ad ogni (ri-)autorizzazione di un MN, la necessità di una relocation ad esempio secondo le politiche precedentemente esposte. Nel caso in cui sia necessaria una HA relocation, il server AAA dell'MSP notifica l'inizio della procedura

di relocation all'HA attivo. L'MSP comunica inoltre all'HA attivo anche l'indirizzo dell'HA subentrante affinché sia inoltrato al MN con un messaggio di HA switch (descritto in [27]) in modo tale che il MN possa registrarsi presso il nuovo HA.

Qualora il nuovo HA si trovasse in una sottorete diversa da quella dell'HA attivo il MN dovrà configurare un nuovo Home Address ed utilizzare il vecchio HoA per le sessioni attive ed il nuovo HoA per le sessioni successive all'HA relocation.

4.6 Aspetti di sicurezza nella separazione tra autenticazione ed autorizzazione

La separazione delle funzionalità di autenticazione ed autorizzazione può comportare l'utilizzo di server diversi (AAAH-EAP e AAAH-MIP6) a causa del routing Diameter basato sia sul dominio che sull'Application-Id dei messaggi (cfr. §3.1.5). Questo porta a considerare una questione: separare la fase di autenticazione da quella di autorizzazione utilizzando applicazioni diverse per le due fasi può comportare un problema di sicurezza?

Sulla mailing list del DIME è stato descritto tto il caso di un HA che viene compromesso da un utente malevolo, il quale può così ottenere l'autorizzazione all'utilizzo del servizio MIPv6 ad esempio impersonando altri utenti. È stato proposto di risolvere questo problema attraverso l'utilizzo di un cosiddetto *authentication token*, a conferma dell'avvenuta autenticazione da parte di un server di autenticazione (ad esempio AAAH-EAP).

L'utilizzo di un authentication token, però, complica le cose: tale certificazione dovrebbe essere firmata dal server AAAH-EAP, comunicata all'HA autenticatore e consegnata al server AAAH-MIP6 delle autorizzazioni. Dovrebbe essere così predisposta una comunicazione tra AAAH-EAP e AAAH-MIP6 per lo scambio di chiavi e segreti condivisi,

Pertanto è bene fare un passo indietro ed analizzare se il problema della conferma dell'autenticazione sia realmente un problema nella soluzione proposta per l'interfaccia HA-AAA.

In realtà, la vulnerabilità causata da un HA compromesso esiste anche in soluzioni che non separano autenticazione e autorizzazione: un utente malintenzionato che riesca a controllare un HA può usufruire il servizio al quale non è autorizzato semplicemente evitando le operazioni di autenticazione/autorizzazione.

Pertanto se il problema di sicurezza si presenta solo nella eventualità che un HA venga compromesso, la separazione delle funzionalità di autenticazione ed autorizzazione non sembra giustificare ulteriori mecansimi di sicurezza come l'authentication token.

Capitolo 5
Lavoro di sviluppo

5.1 Descrizione del testbed

Per l'implementazione dell'interfaccia HA-AAA abbiamo allestito un testbed presso il laboratorio di riferimento della struttura T.IE.CS.CI di Telecom Italia. Il trial è stato realizzato utilizzando un nodo mobile, un Home Agent ed un server AAA collocati in tre diverse sottoreti (realizzate tramite opportune VLAN) interconnesse con un router IPv6, come mostrato in Figura 5.1. Il numero associato alle VLAN rappresentate è solo esemplificativo e non rispecchia l'effettiva impostazione presente nella rete di laboratorio.

Come nodo mobile abbiamo utilizzato un laptop, mentre sono state allestite due macchine virtuali (gestite da un server ESX VMware), una per realizzare il server AAA ed una per l'Home Agent. Come sistema operativo abbiamo utilizzato su tutte le macchine la distribuzione Linux Ubuntu 6.06 LTS (Long Term Support).

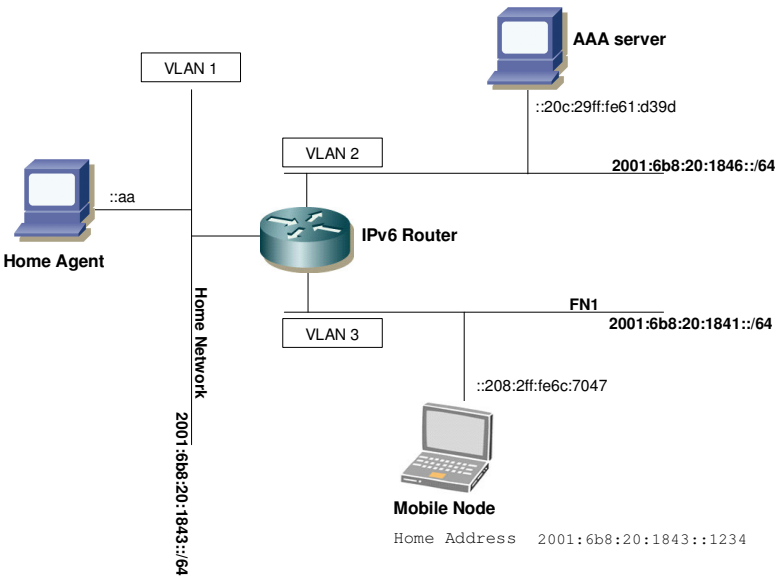

Figura 5.1 – Rappresentazione schematica del testbed

Le componenti software utilizzate sono state Mobile IPv6 per Linux (MIPL) ed Open Diameter. MIPL implementa il protocollo Mobile IPv6 ed è stato installato su MN e HA. Open Diameter implementa il protocollo DIAMETER base ed alcune applicazioni (ad es. EAP e NASREQ) ed è stato utilizzato per l'HA, ovvero il client Diameter, e per il server AAA.

Per questo testbed non abbiamo previsto alcuna autenticazione all'accesso per il nodo mobile. Inoltre non abbiamo utilizzato IPsec per la segnalazione tra HA e MN, facendo l'assunzione che il MN fosse già autenticato.

5.2 Installazione del software

5.2.1 MIPL

Mobile IPv6 per Linux (MIPL) [28] è un'implementazione software open source in linguaggio C del protocollo Mobile IPv6 disponibile sotto licenza GNU GPL. Il codice originario risale ad un progetto del corso di Software Project presso la Helsinki University of Technology (HUT).

Fino alla release 1.1 (per kernel Linux serie 2.4) l'implementazione di MIPv6 era completamente integrata nel kernel, ma dalla versione 2.0 (serie 2.6 del kernel) la maggior parte delle funzionalità sono implementate nello user space.

La versione da noi utilizzata è la 2.0.2 che richiede l'utilizzo della versione 2.6.16 del kernel previa applicazione di una patch per l'abilitazione delle necessarie funzionalità per MIPv6. I file necessari all'installazione di MIPL sono reperibili all'indirizzo [28]:

- `mipv6-2.0.2.tar.gz` (user space);
- `mipv6-2.0.2-linux-2.6.16.patch.gz` (patch MIPL per il kernel 2.6.16);
- `linux-2.6.16.tar.gz` (kernel linux cui applicare la patch).

Dopo aver provveduto all'estrazione dell'archivio del kernel ed esserci spostati nella directory principale del kernel abbiamo testato la patch con il comando

```
zcat PATH/mipv6-2.0.2-linux-2.6.16.patch.gz | patch
-p1 --dry-run
```

e successivamente, in assenza di errori, abbiamo effettivamente applicato la

patch con il comando

```
zcat /_directory_/mipv6-2.0.2-linux-2.6.16.patch.gz
| patch -p1
```

A questo punto dopo aver copiato il file `.config` contenente la precedente configurazione del sistema (presente in `/boot/config-2.6.15-23-server` nel nostro caso) nella directory principale del kernel abbiamo lanciato il comando

```
make oldconfig
```

In questo modo tutte le opzioni del vecchio kernel sono conservate lasciando all'utente il compito di configurare solo le (eventuali) nuove opzioni disponibili per il nuovo kernel.

Una volta completata questa operazione, la nuova configurazione è stata verificata utilizzando lo script

```
./chkconf_kernel.sh <PATH_KERNEL>
```

anch'esso presente nella distribuzione della MIPL. L'output prodotto dallo script mostra se tutte le opzioni relative a IPv6 necessarie a MIPL sono correttamente configurate. Riportiamo il dettaglio delle opzioni presenti nel file `INSTALL.kernel` contenuto nella distribuzione della MIPL:

```
CONFIG_EXPERIMENTAL=y
CONFIG_SYSVIPC=y
CONFIG_PROC_FS=y
CONFIG_NET=y
CONFIG_INET=y
CONFIG_IPV6=y
CONFIG_IPV6_MIP6=y
CONFIG_XFRM=y
CONFIG_XFRM_USER=y
CONFIG_XFRM_ENHANCEMENT=y
* The Home Agent and Mobile Node also need:
CONFIG_IPV6_TUNNEL=y
CONFIG_IPV6_ADVANCED_ROUTER=y
```

```
CONFIG_IPV6_MULTIPLE_TABLES=y
* The Mobile Node also needs:
CONFIG_IPV6_SUBTREES=y
* For some additional movement indicators on the Mobile
  Node you may set:
CONFIG_ARPD=y
* For IPsec support you need at least:
CONFIG_INET6_ESP=y
* If you plan to use IPsec tunnel mode you need:
CONFIG_NET_KEY=y
CONFIG_NET_KEY_MIGRATE=y
```

Modificate queste voci nel file .config abbiamo infine ricompilato il kernel, operazione i cui comandi base sono:

```
make bzImage
make modules
make modules_install
```

Successivamente si è compilata e installata (utilizzando le istruzioni del file `INSTALL`) la parte user space, costituita dal demone di gestione della mobilità *mip6d*. I comandi utilizzati sono stati:

```
CPPFLAGS='-isystem <PATH_HEADERS>' ./configure
```

dove PATH_HEADERS è la directory contenente gli header del kernel ricompilato, e

```
make && make install
```

Al momento dell'esecuzione del demone mip6d è necessario passare da riga di comando il file di configurazione. Nella directory `extras` della distribuzione MIPL sono presenti file di esempio per differenti configurazioni. Di seguito elenchiamo le configurazione utilizzata per il MN e per l'HA.

5.2.1.1. Configurazione Mobile Node

Riportiamo gli elementi più significativi del file di configurazione utilizzato sul MN:

```
NodeConfig MN;

DebugLevel 10;

DoRouteOptimizationCN disabled;

DoRouteOptimizationMN disabled;

Interface "eth0";

MnHomeLink "eth0" {
        HomeAgentAddress 2001:6b8:20:1843::aa;
        HomeAddress 2001:6b8:20:1843::1234/64;
}

UseMnHaIPsec disabled;
```

Figura 5.2 – Esempio di configurazione per un MN MIPL

Con questa configurazione l'host opera come un MN e il demone MIPL stampa a video i messaggi di debug e controllo generati durante l'esecuzione, a causa del valore `DebugLevel` maggiore di 0. La funzionalità di Route Optimization è disabilitata sia con i Correspondent Node sia con altri Mobile Node. L'interfaccia abilitata al servizio di mobilità è eth0 e relativamente a tale interfaccia sono configurati l'indirizzo dell'Home Agent e l'Home Address. Infine l'utilizzo di IPsec è stato disabilitato (IPsec non è stato utilizzato in quanto accessorio ai fini del lavoro di sviluppo).

5.2.1.2. Configurazione Home Agent

Riportiamo gli elementi più significativi del file di configurazione utilizzato sull'HA:

```
NodeConfig HA;

DebugLevel 10;

Interface "eth1";

UseMnHaIPsec disabled;
```

Figura 5.3 – Esempio di configurazione per un HA MIPL

Con tale configurazione l'host opera come un Home Agent e il demone stampa a video i messaggi di debug e controllo generati durante l'esecuzione. Inoltre, l'interfaccia abilitata al servizio di mobilità è eth1 (quest'opzione risulta utile qualora l'HA fosse multihomed). Infine l'utilizzo di IPsec nella comunicazione con nodi mobili risulta disabilitato.

5.2.2 Open Diameter

Open Diameter [29] è un applicativo realizzato in C++ e distribuito con licenza GNU che implementa il protocollo Diameter base ed alcune applicazioni Diameter. Open Diameter non è un software molto maturo: nell'implementazione utilizzata (release 1.0.7-h) è supportata solamente l'applicazione Diameter EAP, ma sono presenti anche prototipi, sotto forma di eseguibili di test con le rispettive librerie, per le applicazioni Diameter NASREQ e MIP4. Al momento della pubblicazione di questa tesi di laurea segnaliamo la disponibilità della nuova versione 1.0.7-i, per la quale si

rimanda a [30].

5.2.2.1. Setup

Open Diameter è supportato da Linux, BSD e Windows, grazie all'utilizzo della libreria ACE (Adaptive Communication Environment, reperibile da [31]), che permette di astrarre il sistema operativo sottostante, e delle librerie BOOST [32] per funzionalità di parser generator.

Prima di poter compilare Open Diameter è necessario verificare la presenza dei seguenti pacchetti Ubuntu (in caso contrario è necessario scaricarli ed installarli ad esempio tramite `apt-get`):

```
- gcc                  (4:4.0.3-1)
- g++-4.0              (4.0.3-1ubuntu5)
- automake1.9
- libtool              (1.5.22-2)
- openssl              (0.9.8a-7build1)
- libssl-dev           (0.9.8a-7build1)
- libboost-dev         (1.33.1-2)
- libxerces26c2        (2.6.0-6)
- libxerces26-dev
```

Riguardo ad ACE, non essendo disponibili pacchetti compilati, è necessario procedere alla compilazione dai sorgenti preventivamente estratti dall'archivio `ACE-5.5.tar.gz`, abilitando il supporto per IPv6 e SSL. A tal fine, è necessario modificare il file `config.h` nella directory `../build/ace` ponendo ad 1 l'attributo ACE_HAS_IPV6, quindi compilare i sorgenti con il comando

```
make ssl=1
```

Una volta completata l'installazione di ACE, si può procedere con l'installazione di Open Diameter. Dopo aver estratto l'archivio

`opendiameter-1.0.7-h.tar.gz`, è necessario procedure alla compilazione lanciando il comando

```
make
```

Prima di poter utilizzare Open Diameter bisogna accertarsi di settare le seguenti variabili di ambiente attraverso i seguenti comandi bash:

```
export ACE_ROOT=PATH/ACE_wrappers;
export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib:$AC
E_ROOT/lib:$ACE_ROOT/ace;
export XERCESCROOT=/usr/include;
export BOOST_ROOT=/usr/include;
```

avendo cura di sostituire nella prima riga PATH con il nome della directory radice di `ACE_wrappers`.

È opportuno inserire tali righe di codice nel file `/root/.bash_profile`, per evitare di ripetere questa procedura ad ogni riavvio del server.

5.3 Implementazione dell'applicazione Diameter MIP6 Authorization

Il lavoro di sviluppo è stato preceduto dalla decisone sull'approccio da seguire per la creazione della nuova applicazione MIP6 Authorization. Dopo un'analisi dell'architettura software di Open Diameter (nella versione 1.0.7-h da noi utilizzata), si sono presentate due possibilità:

1) utilizzare come base di partenza alcuni applicativi di test client e server presenti nella libreria `/libdiameter`;

2) realizzare una libreria ad hoc per l'applicazione alla stregua di quelle presenti per EAP e NASREQ.

La scelta è caduta sulla seconda opzione e, in particolare come base di partenza si è deciso di utilizzare la libreria relativa all'applicazione NASREQ giudicata la più idonea per i seguenti motivi: la migliore organizzazione del codice sorgente e la volontà di creare una soluzione in grado di essere più facilmente integrabile nel pacchetto Open Diameter.

Il lavoro di sviluppo ha portato alla creazione di un prototipo funzionante di applicazione MIP6 Authorization racchiuso in una directory chiamata `libdiametermip6/` che implementa un sottoinsieme delle funzionalità definite in §4.5. In particolare, abbiamo realizzato:

- lo scambio di messaggi MAR e MAA per la fase di autorizzazione, ovvero messaggi caratterizzati dal valore 2 (AUTHORIZE_ONLY) nell'Auth-Request-Type AVP e dalla presenza di AVP dedicati al trasporto di parametri quali lo username dell'utente da autorizzare oppure il lifetime associato ad un'autorizzazione avvenuta con successo;
- le funzioni e le macchine a stati per la generazione e gestione dei messaggi MAR e MAA nel client e nel server;
- la creazione di un database degli utenti nel server.
- le strutture dati nel client e nel server per il salvataggio dello username dell'utente da autorizzare;
- l'implementazione della funzionalità di ri-autorizzazione gestita dal client, ovvero della possibilità per il client di inviare messaggi MAR di refresh per l'autorizzazione in scadenza di un utente.

Sono state invece tralasciate funzionalità opzionali come il

supporto per l'HA relocation.

Nel nostro trial abbiamo assegnato all'applicazione MIP6 Authorization un Application-ID pari a 6000 e un command code pari a 600 per i messaggi MAR/MAA.

Dettagliamo di seguito il lavoro svolto per l'implementazione dell'applicazione MIP6 Authorization.

5.3.1 Scambio di messaggi MAR/MAA

Partendo dall'applicazione NASREQ implementata in Open Diameter, come prima cosa abbiamo posto a 2 (AUTHORIZE_ ONLY) il valore dell'Auth-Request-Type AVP per verificare il funzionamento dell'applicazione con messaggi di sola autorizzazione.

Tale prova ha mostrato l'incapacità dell'applicazione NASREQ di gestire messaggi di tipo AUTHORIZE_ONLY: abbiamo osservato che il server NASREQ non li considera messaggi validi, restituendo un errore di parsing del messaggio.

Per ottenere un corretto processing di tali messaggi, abbiamo provveduto a studiare e modificare la macchina a stati del server; inoltre abbiamo modificato anche la macchina a stati del client per eliminare stati e transizioni inutili alla procedura di autenticazione.

5.3.1.1. Macchina a stati del server

Riportiamo in Figura 5.4 il comportamento della macchina a stati del server Diameter NASREQ definito nel file `libdiameter nasreq/src/diameter_nasreq_server_fsm.cxx`: le etichette del tipo "Evento / *azione*" sono riferite ai cambi di stato e indicati con una linea

continua, mentre una linea tratteggiata sta ad indicare un *wildcard event*, ovvero un generico evento che comporta un cambio di stato forzoso.

La soluzione implementata prevede che il server consideri validi solamente i messaggi di tipo AUTHORIZE_ONLY. Non è prevista la gestione al server di richieste di altro tipo (AUTHENTICATE_ONLY o AUTHORIZE_AUTHENTICATE).

Dalla Figura 5.4 si evince come non sia prevista una transizione dallo stato `CheckAA_Request` allo stato `WaitAuthorization`, transizione come ci si attenderebbe per il processing di un messaggio di tipo AUTHORIZE_ONLY. Ma i controlli che verificano l'identità dell'utente sono effettuati nello stato `WaitAuthInfo` (con la chiamata della funzione `Validate()`, cfr §5.3.2), ovvero nel momento in cui avviene l'autenticazione dell'utente.

Pertanto abbiamo modificato il nome dello stato `WaitAuthInfo` in `WaitAuthzInfo` demandando a questo stato le verifiche sull'identità e sull'autorizzazione degli utenti (effettuate dalla funzione `Validate()`). La macchina a stati finale è pertanto molto simile a quella dell'applicazione NASREQ a meno dell'eliminazione dello stato `WaitAA_Request`, il quale era previsto per i metodi di autenticazione che necessitano di più round di tipo request/response, e della coppia Evento/Azione che causava la transizione in quello stato.

In Figura 5.5 abbiamo rappresentato la macchina a stati del server per l'applicazione MIP6 Authorization.

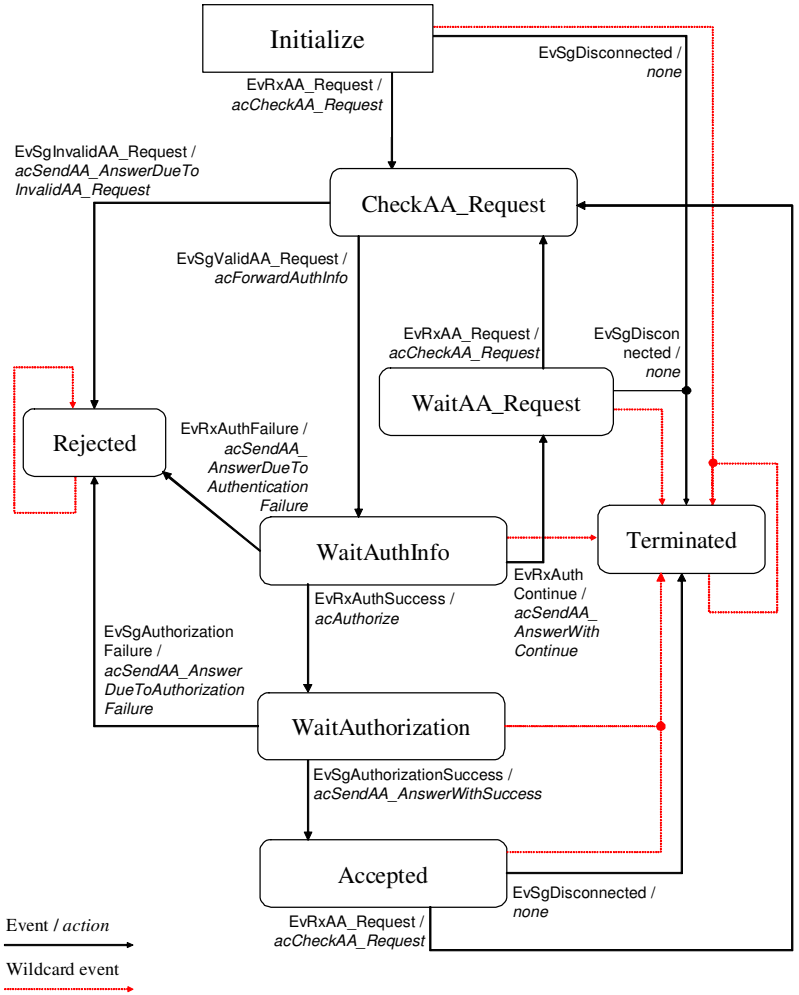

Figura 5.4 – Macchina a stati del server AAA (applicazione NASREQ)

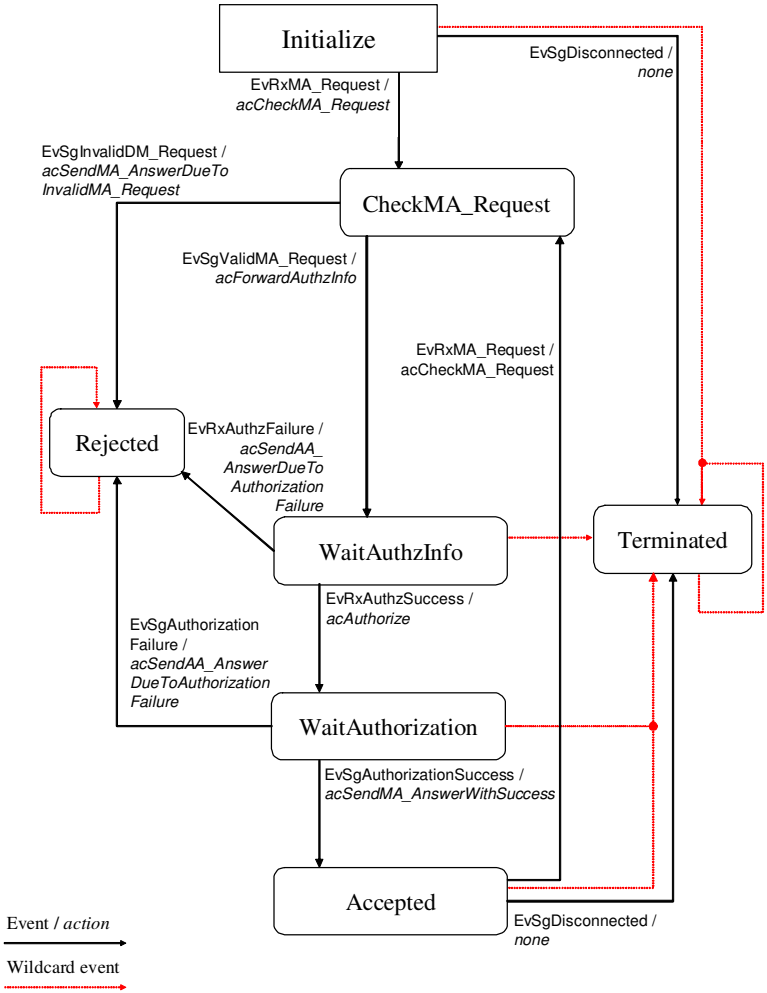

Figura 5.5 – Macchina a stati del server AAA (applicazione MIP6 Authorization)

5.3.1.2. Macchina a stati del client

Anche in questo caso, il punto di partenza è stato la macchina a

stati relativa all'applicazione NASREQ. Il client NASREQ agisce secondo lo schema di Figura 5.6, nel quale si nota la presenza dello stato `WaitAuthInfo`, inutile ai fini dell'autorizzazione come già visto parlando del server NASREQ (cfr. §5.3).

La macchina a stati del client è stata così modificata eliminando lo stato `WaitAuthInfo` e le coppie evento/azione associate, lasciando inalterata la restante struttura generale. La macchina a stati risultante è mostrata in Figura 5.7.

Abbiamo previsto nel client una transizione dallo stato `Accepted` allo stato `Initialize` a seguito della scadenza del lifetime dell'autorizzazione: questa transizione permette al client di inviare messaggi per la ri-autorizzazione di un utente. Per le problematiche relative a questa funzionalità rimandiamo al §5.3.5.

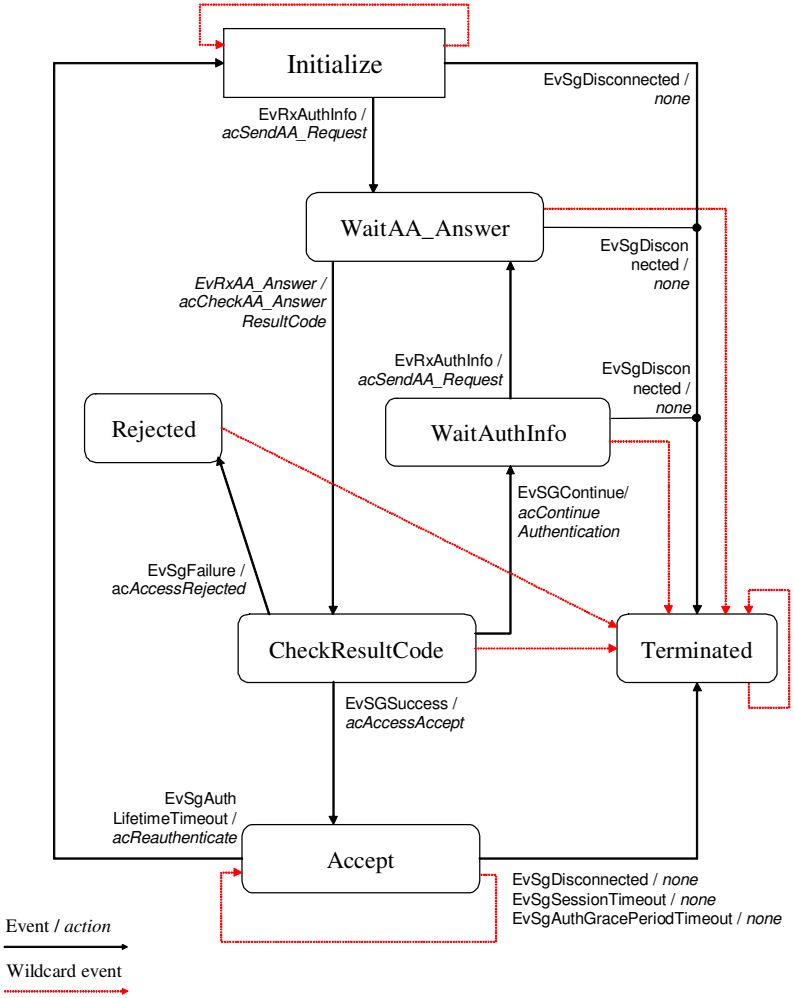

Figura 5.6 – Macchina a stati del client (applicazione NASREQ)

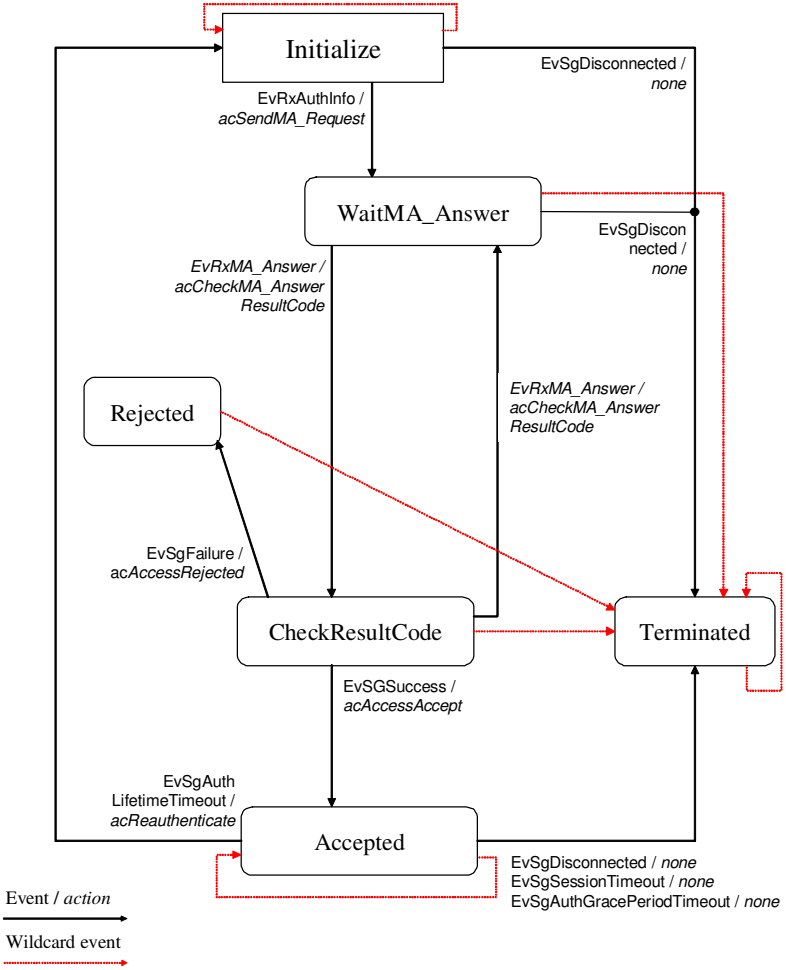

Figura 5.7 – Macchina a stati del client (applicazione MIP6 Authorization)

5.3.2 Database degli utenti

Open Diameter non supporta nativamente database "esterni" quali ad esempio SQL o LDAP: così per la realizzazione del database utenti abbiamo optato per un'opzione realisticamente percorribile, ovvero l'utilizzo di un file XML. Il motivo risiede nel supporto in Open Diameter di librerie per il parse di file XML.

Il database degli utenti è stato implementato attraverso un file denominato `aaa_user_db.xml`. contenente gli username degli utenti abilitati ad usufruire del servizio MIPv6 e la validità temporale espressa in secondi dell'autorizzazione di un particolare utente.

Riportiamo un file di esempio relativo al database degli utenti:

```
<?xml version="1.0" encoding="UTF-8"?>
<user_db xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xsi:noNamespaceSchemaLocation='aaa_user_db.xsd'>

<user_entry>
   <name_match>luca@telecomitalia.it</name_match>
   <authz_lifetime>8</authz_lifetime>
</user_entry>
<user_entry>
   <name_match>michele@telecomitalia.it</name_match>
   <authz_lifetime>20</authz_lifetime>
</user_entry>
</user_db>
```

In questo caso gli utenti autorizzati al servizio MIPv6 sono *luca* e *michele*, ai quali sono associati rispettivamente valori di lifetime pari a 8 e 20 secondi.

A seguito dell'interrogazione del database, i dati sono inseriti in

strutture dati implementate nel file `libdiametermip6/include/diameter_mip6_authinfo.hxx` creato sul modello del file `diameter_nasreq_authinfo.hxx`, contenente le classi per le informazioni di autenticazione per l'applicazione NASREQ.

La classe creata per il salvataggio dello username e del lifetime associato ad un utente è stata denominata `USER_Info`. Per tale classe sono state definite le variabili private `User` e `lifetime` le funzioni membro pubbliche necessarie al recupero dei dati contenuti nelle variabili medesime.

Inoltre nella classe `USER_Info` abbiamo inserito la funzione membro pubblica `Validate()`, prendendo spunto dalla funzione omonima della classe `PAP_Info` dell'applicazione NASREQ. La funzione `Validate()` consulta il database degli utenti `aaa_user_db.xml`: se lo username per il quale viene fatta la richiesta di autorizzazione è presente nel database la funzione restituisce il valore booleano "vero" e pone la variabile privata `lifetime` al valore associato all'utente; in caso contrario, restituisce il valore "falso" causando il fallimento della richiesta di autorizzazione, ovvero l'invio di un messaggio MAA con Result-Code 5003 (`DIAMETER_AUTHORIZATION_REJECTED`).

Il file `diameter_mip6_authinfo.hxx` è allegato nel paragrafo D.2.4 in Appendice D.

5.3.3 Interazione tra client Diameter e demone MIPv6

Il client Diameter deve potersi interfacciare con il demone (mip6d) che gestisce la mobilità sull'HA (cfr. §4.5). Tale interfaccia MIPL-OpenDiameter è stata realizzata con una socket UDP interna all'Home

Agent.

Dopo aver instaurato una connessione con il server, il client Diameter rimane in ascolto sulla socket UDP (porta 47901) in attesa di messaggi da MIPL che inneschino una sessione di autorizzazione. La struttura dei messaggi provenienti dal demone MIPL è mostrata in Figura 5.8 ed è spiegata di seguito.

Il campo *Type,* su sedici bit, specifica il tipo di messaggio ed i valori ammessi sono due: 1 per Authorization Request (messaggio dal demone MIPL al client Diameter), 2 per Authorization Reply (messaggio dal client Diameter al demone MIPL). Il campo *State,* su sedici bit, è diverso da zero nei messaggi di tipo Reply e contiene il codice relativo all'esito della verifica dell'autorizzazione: 1 in caso di autorizzazione avvenuta con successo, 0 in caso contrario.

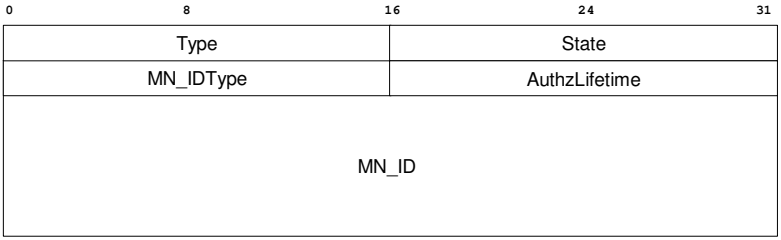

Figura 5.8 – Formato del messaggio UDP per la socket interna al client

Il campo *MN_IDType* è anch'esso su sedici bit e specifica il tipo di identità da autorizzare che è contenuta nel campo *MN_ID*. Nella versione attuale esso è predisposto ad accogliere un Network Access Identifier (NAI), al quale è associato un valore di MN_IDType pari ad 1. Il NAI contenuto è nella forma di stringa di caratteri terminata dal carattere di fine

stringa "\0", motivo per cui non è stato necessario definire un campo contenente la lunghezza dell'ID utente. Il campo *MN_IDType* è stato previsto per l'estensione in versioni successive ad altri tipi di identificativi, quali ad esempio un Fully Qualified Domain Name (FQDN), un indirizzo e-mail o IPv6 oppure un identificativo specifico di altri protocolli.

Il campo *AuthzLifetime* su sedici bit contiene la durata in secondi del periodo di validità dell'autorizzazione, qualora essa sia andata a buon fine, e può essere diverso da zero solo nei messaggi di tipo Reply.

5.3.4 Sessioni di autorizzazione

Il client Diameter, in ascolto sulla porta 47901, crea una nuova sessione di autorizzazione alla ricezione di un messaggio di tipo Request proveniente dalla socket con il demono MIPL. La classe che gestisce la sessione è chiamato `HA_Application`, presente nel file `test/client_test.cxx` (appendice §D.2.2); il costruttore della classe `HA_Application` riceve come parametro l'identità dell'utente da autorizzare contenuta nel messaggio di Request ricevuto.

Ogni sessione creata alla ricezione di una Request proveniente dal demone MIPL è gestita da un thread indipendente per il quale vengono allocate risorse al momento della creazione e rilasciate alla distruzione (API multithreading di Open Diameter). Per un'approfondimento di queste funzionalità si rimanda a [33] e [34].

Le sessioni di autorizzazione possono essere di tipo stateful o stateless. Per sessioni stateful il server Diameter mantiene lo stato delle autorizzazioni attive: ad esempio a causa di un cambiamento dello stato di un utente attivo (i.e. l'esaurimento del credito per un servizio prepagato), il

server può terminare una sessione stateful ed inviare un messaggio di Abort-Session-Request al client. Le sessioni stateless, invece, non prevedono meccanismi di interazione tra client e server oltre allo scambio richiesta/risposta per l'autorizzazione; questo significa ad esempio che se un utente autorizzato termina il servizio prima della scadenza del lifetime relativo alla sua autorizzazione, il client Diameter non è tenuto a informare il server di questo evento con il messaggio Session-Termination-Request. Operativamente, il comportamento stateless o stateful di un peer può essere predeterminato dal file di configurazione (nel nostro caso il file `client.local.xml` presente nella directory `/config)` oppure può essere richiesto dal client in un messaggio MAR per una specifica sessione con l'inserimento dell'AVP Auth-Session-State con valore 1 (cfr. §3.1.7.14).

Abbiamo verificato sperimentalmente che nel client la durata dell'autorizzazione concessa agli utenti era superiore al valore configurato nel database. Il motivo risiedeva in una erronea implementazione del timer relativo alla durata dell'autorizzazione a livello di API Open Diameter, che illustriamo nel prossimo sottoparagrafo.

5.3.5 Modifiche all'API di Open Diameter

L'API (Application Programming Interface) di Open Diameter [33] è un insieme di classi e funzioni di base utilizzate per sviluppare il protocollo base ma anche le applicazioni Diameter. L'API Open Diameter si compone di sub-API di tipo Job, Task e State Machine per l'implementazione del protocollo Diameter.

Abbiamo riscontrato sia all'interno del file `libdiameter/include/aaa_session_auth_client_fsm.h` sia del file

`libdiameter/include/aaa_session_auth_server_fsm.h` che Authorization Lifetime e Grace Period Lifetime sono gestiti con un unico timer denominato `AAA_TIMER_TYPE_AUTH`. Tale timer viene inizializzato come somma di Authorization Lifetime e Grace Period Lifetime e fatto partire al momento della ricezione di un messaggio MAA valido (ovvero completo di tutti gli AVP necessari all'autorizzazione) e dal Result Code pari a `DIAMETER_SUCCESS`. Questo avviene sia per client stateful che stateless. Nel seguito facciamo riferimento a client stateful senza perdere di generalità, anche se il client utilizzato nel nostro trial è di tipo stateless (si veda §5.4)

Nell'RFC 3588 al paragrafo 8.1 "`Authorization Session State Machine`" (pag. 94) è descritta la macchina a stati di un client stateful per l'autorizzazione che chiarisce le scelte implementative descritte in precedenza:

```
State   Event                     Action     New State
-------------------------------------------------------
Open    Authorization-Lifetime    Send STR   Discon
        + Auth-Grace-Period
        expires on access device
```

come per la macchina a stati del server stateful (pag. 95):

```
State   Event                     Action     New State
-------------------------------------------------------
Open    Authorization-Lifetime    Cleanup    Idle
        (and Auth-Grace-Period)
        expires on home server
```

Una spiegazione di questa implementazione si può trovare nel paragrafo 8.10 "`Auth-Grace-Period AVP`" (pag. 110) dove, parlando del

server, si dice che l'Auth-Grace-Period AVP contiene il numero di secondi che il server Diameter aspetterà successivamente alla scadenza dell'Authorization Lifetime prima di liberare le risorse della relativa sessione.

Per gestire la ri-autorizzazione iniziata dal client come previsto nella nostra macchina a stati dell'applicazione MIP6 Authorization (cfr. Figura 5.7) abbiamo dovuto modificare il comportamento dell'API Open Diameter relativamente alla macchina a stati del client creando un nuovo timer chiamato `AAA_TIMER_TYPE_AUTHZ`. Il timer `AAA_TIMER_TYPE_AUTHZ` viene avviato alla ricezione di un messaggio MAA di autorizzazione ed inizializzato al valore di Authorization Lifetime ricevuto. In questo modo alla scadenza del timer l'API di Open Diameter notifica correttamente alla macchina a stati l'evento *EvSgAuthLifetimeTimeout* grazie al quale il client passa dallo stato `Accepted` allo stato `WaitMA_Answer` ed invia un messaggio MAR di ri-autorizzazione al server.

5.4 Modifiche alla MIPL

L'interazione con il client Diameter ha richiesto alcune modiche sul codice MIPL per l'implementazione della gestione delle procedure di verifica delle autorizzazioni tramite la socket di comunicazione con Open Diameter. Tutte le modifiche apportate sono disattivabili dall'utilizzatore tramite file di configurazione di MIPL.

Esaminiamo ora alcuni aspetti dello sviluppo eseguito sul codice MIPL.

5.4.1 Interfaccia con Open Diameter

Poiché la procedura di autorizzazione è innescata dalla ricezione di una BU da parte Home Agent (cfr. §4.5), abbiamo deciso di inserire il codice relativo alla comunicazione con Open Diameter nella funzione che si occupa del processing di una generica Binding Update ricevuta dall'Home Agent.

Questa funzione è la `ha_recv_bu_worker` definita all'interno del file `src/ha.c`. Il codice è stato inserito dopo la sezione relativa alla Duplicate Address Detection e prima della creazione della binding entry (di tipo home registration) relativa all'utente. Per maggiore chiarezza riportiamo il contenuto del file `ha.c` da riga 777 a riga 794 (il codice preesistente è mostrato in corsivo):

```
bce = bcache_get(out.src, out.dst);
if (!bce) {
    BUG("BCE deleted before DAD completed!");
    status =  IP6_MH_BAS_UNSPECIFIED;
    goto send_nack;
}
/******** Authorization via Open Diameter *********/
dbg("********** AUTHORIZING USER **********\n");
if (verify_authorization()== 1) {
    dbg("Authorization SUCCESS!\n");
}
else {
    dbg("Authorization FAILED!\n");
    status = IP6_MH_BAS_PROHIBIT;
    goto send_nack;
}
/***********************************************/
new = 1;
```

La funzione `verify_authorization()` è definita nel file `authorization.c` nel quale sono definite anche tutte le funzioni relative alla creazione e gestione della socket e dei messaggi scambiati tra MIPL ed

Open Diameter. La funzione `verify_authorization()` invia un messaggio di tipo 1 (Authorization Request) al client Open Diameter attraverso la socket UDP con l'identità (NAI) dell'utente da autorizzare e processa la risposta corrispondente: in caso di successo la funzione restituisce il valore 1 e la funzione `ha_recv_bu_worker` completa la registrazione e l'attivazione del servizio MIPv6 per l'utente; in caso contrario, l'HA interrompe la registrazione dell'utente inviando al MN un messaggio di Binding Acknowledgement dallo status 129 (Administratively Prohibited).

L'identità dell'utente, nella forma di NAI, non è in generale presente nei messaggi di BU: tipicamente l'HA ha a disposizione solo l'Home Address del nodo mobile, ed è necessario un modo per associare il NAI all'Home Address dell'utente da autorizzare al servizio MIPv6.

Una possibilità è data dalla creazione dinamica di una corrispondenza HoA-NAI in fase di autenticazione. Nel nostro prototipo, come detto alla fine del paragrafo §5.1, non abbiamo implementato la fase di autenticazione che precede l'autorizzazione. Pertanto abbiamo definito staticamente un database implementato con un file di testo chiamato HoA.txt. In questo file abbiamo inserito in due colonne gli Home Address validi e i NAI corrispondenti. Questo database è interrogato dalla funzione `retrieve_NAI()`, definita anch'essa nel file `authorization.c`, che riceve in ingresso l'HoA del nodo mobile e restituisce, se presente, il NAI associato a quell'HoA. La funzione `retrieve_NAI()` è chiamata all'interno della funzione `verify_authorization()`. Di seguito mostriamo un esempio di file HoA.txt.

```
2001:6b8:20:1844:0:0:0:1231
2001:6b8:20:1844:0:0:0:1232   michele@telecomitalia.it
2001:6b8:20:1844:0:0:0:1233
2001:6b8:20:1844:0:0:0:1234   luca@telecomitalia.it
2001:6b8:20:1844:0:0:0:1235
```

Figura 5.9 – Esempio di file HoA.txt

5.4.2 Gestione delle ri-autorizzazioni

MIPL richiede una ri-autorizzazione per un utente attivo registrato (quindi già autorizzato in precedenza) nella Binding Cache alla scadenza del lifetime di autorizzazione. Ricordiamo che il lifetime è comunicato nel messaggio di Authorization Reply inviato dal client Diameter al demone mip6d attraverso la socket UDP interna.

Per la gestione delle ri-autorizzazioni abbiamo implementato in MIPL tre nuovi campi nella struttura della Binding Cache entry, che ricordiamo essere la struttura dati in cui è presente il binding HoA-CoA per un nodo mobile registrato presso l'HA. Tali campi sono: lo username associato all'HoA, recuperato tramite la funzione `retrieve_NAI()` dal file HoA.txt; la durata dell'autorizzazione (lifetime) comunicata dal client; la struttura `authz_tqe` di tipo *timer_queue* che gestisce il timer collegato al lifetime di autorizzazione. Queste modifiche sono state inserite nel file `/src/bcache.h`.

Il valore di lifetime per l'autorizzazione comunicato dal client Diameter è utilizzato per l'inizializzazione; tale timer viene fatto partire al momento dell'inizio dell'erogazione del servizio MIPv6 per un dato utente. Alla scadenza del timer `authz_tqe` ,il demone MIPL triggera una ri-

autorizzazione per l'utente inviando un messaggio di Authorization Request nella socket UDP al client Open Diameter con lo username dell'utente da ri-autorizzare. La procedura è identica a quella di una prima autorizzazione. Il client Diameter, di conseguenza, inizia la ri-autorizzazione dell'utente inviando un messaggio MAR al server con lo username comunicato dal demone MIPL e, in caso di ricezione di un messaggio MAA di avvenuta autorizzazione, invia al demone un messaggio di successo con il nuovo valore di lifetime per l'utente.

5.4.3 Attivazione delle nuove funzionalità

Un'utile funzionalità implementata è la possibilità di abilitare o disabilitare le funzionalità relative all'autorizzazione attraverso il file di configurazione di MIPL.

```
NodeConfig HA;
DebugLevel 10;
Interface "eth1";
UseMnHaIPsec disabled;
HaveAuthorization enabled;
HoAsFile "/etc/HoA.txt";
```

Figura 5.10 – Esempio di file di configurazione con le nuove funzionalità di MIPL

La MIPL utilizza l'applicativo GNU *bison* [35] per generare automaticamente il codice (file `src/scan.c`) per il parsing del file di configurazione. Pertanto, per implementare due nuovi identificatori (*token*) per il file di configurazione e le regole ad essi relative abbiamo agito sui file

`/src/scan.l` e `/src/gram.y` utilizzati da bison.

Tali token sono `HaveAuthorization` e `HoAsFile`. il primo seguito da *enabled* o *disabled* abilita/disabilita le funzionalità di autorizzazione da noi sviluppate ed il secondo, seguito da un path, indica il path completo del file HoA.txt che contiene la corrispondenza Home Address-NAI degli utenti.

Per maggiore chiarezza abbiamo riportato un file di configurazione di esempio con i nuovi token in Figura 5.10.

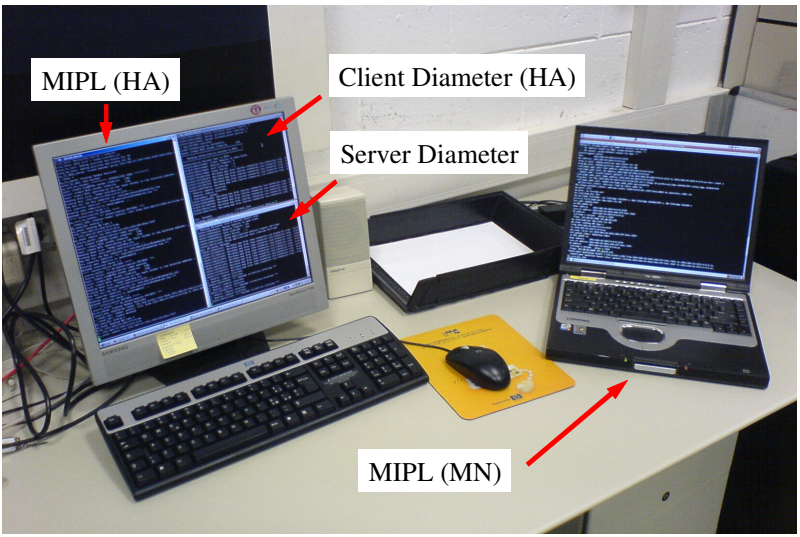

Figura 5.11 – Presentazione visiva del testbed

5.5 Risultati

Mostriamo ora alcune figure relative al testbed e ad alcuni snapshot

di messaggi prodotti dagli eseguibili MIPL ed Open Diameter.

In Figura 5.11 abbiamo raffigurato la postazione utilizzata per il lavoro di sviluppo. Il pc fisso accede in remoto alle macchine virtuali utilizzate come Home Agent e server AAA, ed ognuna delle finestre di terminale mostra un eseguibile in esecuzione: il demone *mip6d* della MIPL sull'Home Agent e gli eseguibili di Open Diameter *client_test* sull'HA e *server_test* sul server AAA. Il nodo mobile esegue anch'egli il demone *mip6d* della MIPL.

Riportiamo un'autorizzazione avvenuta con successo per un utente mobile. Per semplicità di lettura riportiamo in Figura 5.12 il message flow di una procedura di autorizzazione:

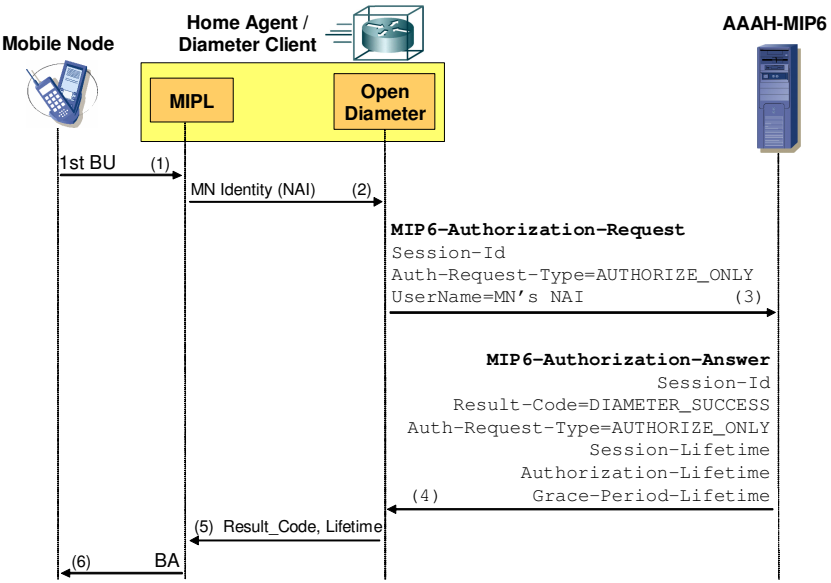

Figura 5.12 – Message flow per l'autorizzazione

All'arrivo di una BU richiedente la registrazione di un nuovo utente (step 1 di Figura 5.13), l'HA verifica l'autorizzazione dell'utente: apre la socket con Open Diameter, consulta il file degli HoA, recupera (se presente) il NAI associato all'HoA presente nella BU e invia un messaggio di Authorization Request sulla socket verso il client Diameter (step 2 di Figura 5.13).

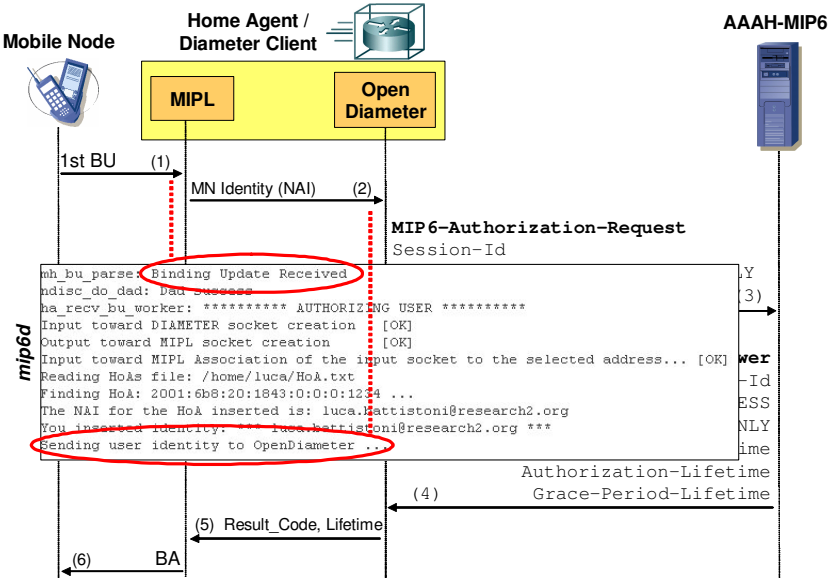

Figura 5.13 – Snapshot n° 1 (MIPL sull'HA)

Il client Open Diameter riceve il messaggio di Authorization Request spedito da MIPL nella socket UDP interna (step 2 di Figura 5.14) invia un messaggio MAR (step 3 di Figura 5.14) per verificare

l'autorizzazione dello username comunicato da MIPL.

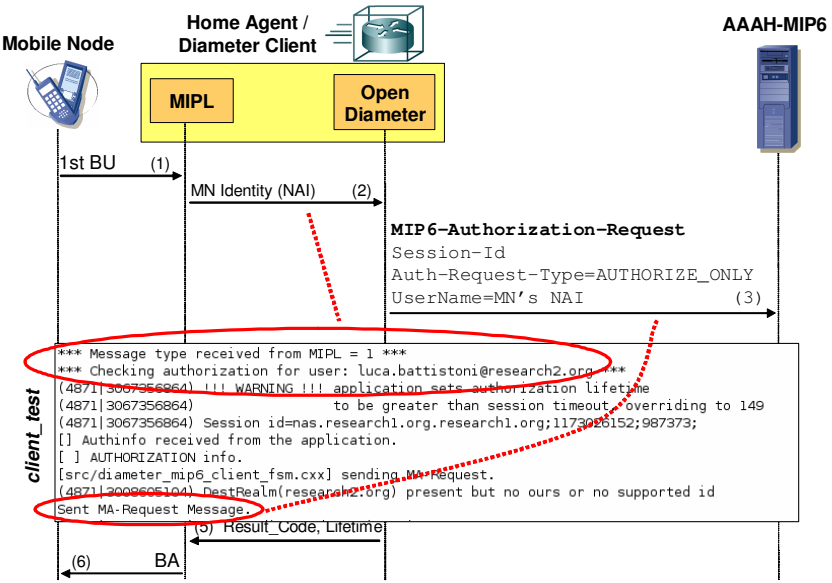

Figura 5.14 – Snapshot n° 2 (client Diameter sull'HA)

Il server Diameter riceve la richiesta di autorizzazione (passo 1 in Figura 5.15) e controlla la presenza dello username ricevuto nel database XML degli utenti (passo 2 e 3). Poiché lo username è presente nel database, l'utente è autorizzato al servizio Mobile IPv6 e la richiesta ha successo (passo 4). Il server prepara un messaggio MAA nel quale inserisce nell'Authorization-Lifetime AVP il valore presente nel database, in questo esempio pari ad 8 secondi (passo 5).

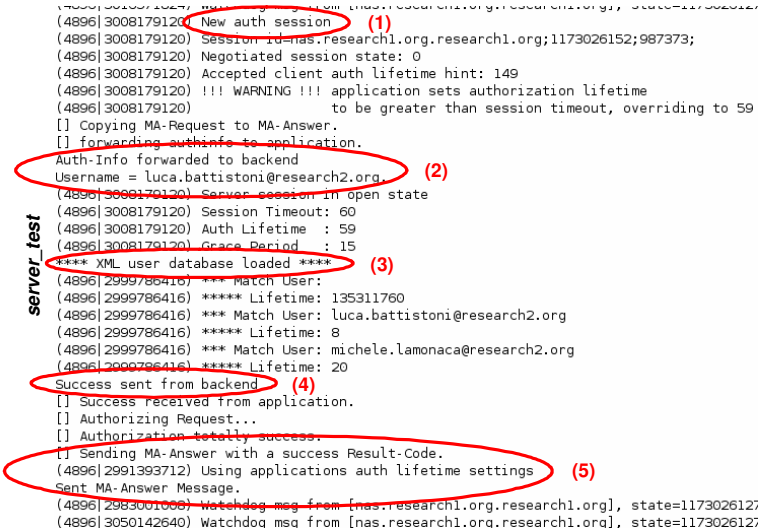

Figura 5.15 – Snapshot n° 3 (server Diameter)

Il client Diameter riceve il messaggio MAA, constata il successo dell'operazione, recupera il valore di lifetime inviato dal server (step 4 di Figura 5.16) e comunica l'autorizzazione avvenuta con successo alla MIPL (step 5 di Figura 5.16).

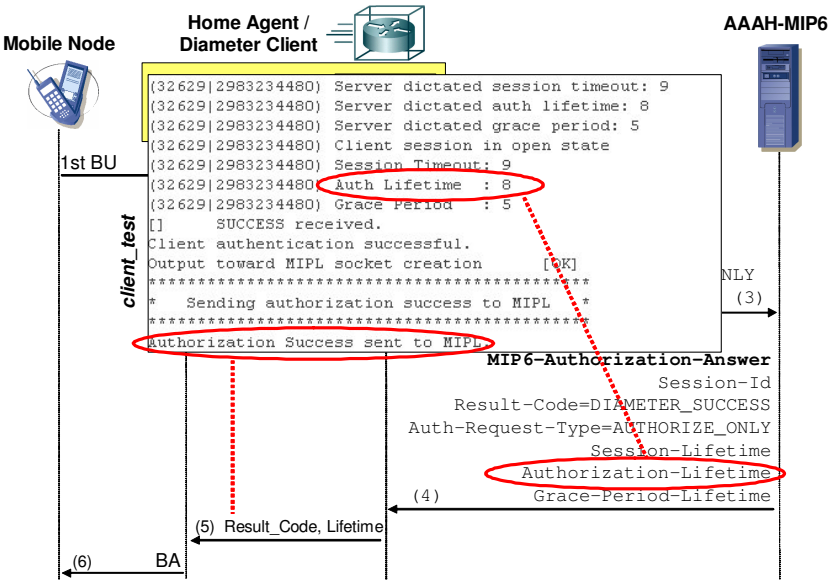

Figura 5.16 – Snapshot n° 4 (client Diameter sull'HA)

Il demone mip6d riceve il messaggio di Authorization Answer dalla scoket UDP (messaggio di tipo 2 evidenziato in Figura 5.17) e completa la registrazione del nodo mobile, inviando una Binding Acknowledgement (step 6 di Figura 5.17).

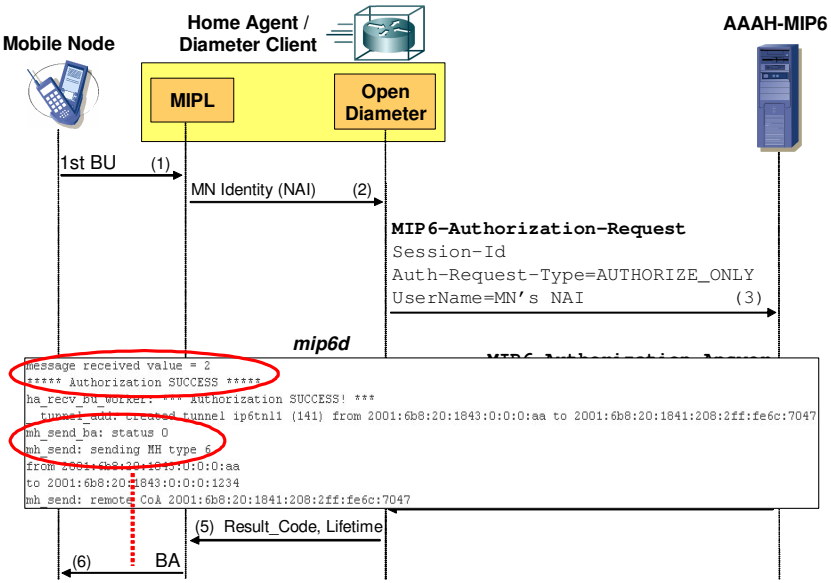

Figura 5.17 – Snapshot n° 5 (MIPL sull'HA)

Conclusioni

In questo lavoro di tesi abbiamo progettato e sviluppato un'interfaccia tra HA e server AAA per l'autorizzazione del servizio Mobile IPv6 (MIPv6).

Inizialmente abbiamo studiato lo stato dell'arte in IETF attraverso i draft pubblicati dai gruppi di lavoro MIP6 e DIME. Abbiamo poi valutato i requisiti necessari per l'interazione tra il mondo MIPv6 e la piattaforma AAA di un operatore, contestualizzando l'interfaccia HA-AAA nei diversi scenari del più ampio problema del bootstrap di MIPv6.

La definizione dell'interfaccia si è basata su un documento draft del DIME wg che formalizza la separazione delle fasi di autenticazione e di autorizzazione relativamente all'avvio del servizio Mobile IPv6. L'applicazione Diameter prevista nel documento per l'interfaccia HA-AAA è chiamata MIP6 Authorization: tale applicazione lascia aperte alcune problematiche che abbiamo tentato di risolvere con soluzioni originali presentate in questo lavoro di tesi (scelta della ricezione della prima BU quale trigger della MIP6 Authorization, supporto per autenticazioni di tipo

IKE-PSK e Authentication Protocol, supporto per la HA relocation).

Una volta definita l'interfaccia, siamo passati allo sviluppo di un prototipo funzionante dell'applicazione utilizzando un testbed predisposto ad-hoc presso i laboratori Tilab di Telecom Italia: dopo aver progettato gli algoritmi, i messaggi e i parametri necessari all'implementazione, abbiamo realizzato una versione "light" dell'applicazione MIP6 Authorization utilizzando le suite MIPL (Mobile IPv6 for Linux) e Open Diameter facendole interagire tra di loro.

Abbiamo implementato un pacchetto composto da un'applicativo di test e dalle relative librerie, denominato libdiametermip6, nell'ottica di una futura integrazione con Open Diameter.

Nell'HA abbiamo realizzato una versione "custom" di MIPL che si interfaccia con Open Diameter attraverso una socket interna all'HA: tale canale di comunicazione è utilizzato per l'interscambio di informazioni e parametri relativi all'autorizzazione. Abbiamo inoltre previsto che sia la stessa MIPL a triggerare il client Diameter per la gestione delle riautorizzazioni. Infine, abbiamo avuto cura che le nuove funzionalità di interlavoro con Open Diameter possano essere facilmente abilitate/disabilitate attraverso il file di configurazione di MIPL.

Appendice A
IPv6

A.1 Introduzione ed indirizzi

IPv6 rappresenta la versione 6 dell'Internet Protocol. IPv6 introduce alcuni nuovi servizi e semplifica molto la configurazione e la gestione delle reti IP. La sua caratteristica principale, però, è il più ampio spazio di indirizzamento: IPv6 gestisce fino a circa $3{,}4 \times 10^{38}$ indirizzi (280.000.000.000.000.000 di indirizzi unici per ogni metro quadrato della superficie terrestre), mentre IPv4 gestisce soltanto fino a circa 4 miliardi (4×10^{9}) di indirizzi.

Il 20 luglio 2004 l'ICANN ha annunciato che i root server DNS erano stati modificati per supportare sia il protocollo IPv6 che IPv4. Si pensa che il protocollo IPv4 verrà utilizzato fino al 2025 circa, per dare il tempo necessario a correggere gli eventuali errori[1].

[1] fonte: Wikipedia - http://it.wikipedia.org/wiki/IPv6

A.2 Tipologie di indirizzi

Un indirizzo IPv6 è rappresentato su 128 bit, tipicamente raggruppati in 8 gruppi da 16 bit in notazione esadecimale separati dal carattere ":". Mostriamo un esempio di indirizzo IPv6:

```
133:abc3:d331:90c::32:ad
```

Il carattere jolly "::" può essere utilizzato al più una sola volta per indicare la presenza di gruppi di soli zeri. Ad esempio, l'indirizzo proposto qui sopra per esteso risulta essere

```
133:abc3:d331:90c:0:0:32:ad
```

IPv6 introduce tre tipologie di indirizzi: indirizzi unicast, multicast ed anycast. Gli indirizzi unicast identificano una singola interfaccia di rete, e si caratterizzano per il concetto di visibilità secondo l'ambito (*scope*) di utilizzo: esistono gli indirizzi *link-local*, caratterizzati dal prefisso `0xfec0::/10`, che limitano la visibilità dei pacchetti al link fisico dell'interfaccia (i router non inoltrano pacchetti con indirizzi link-local); e indirizzi *global* visibili sull'intera Internet, che sono caratterizzati dai primi tre bit posti al valore `001`.

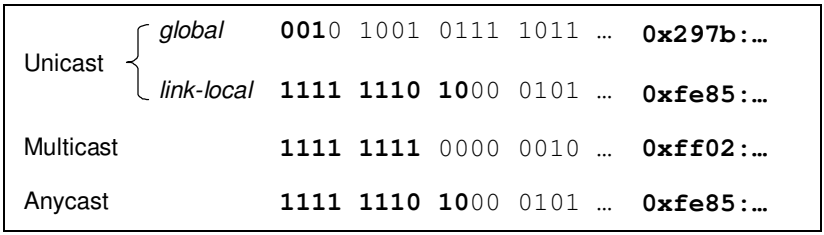

Figura A.1 – Esempi delle tipologie di indirizzi IPv6

Gli indirizzi multicast identificano più di una interfaccia di rete. Un indirizzo multicast definisce un cosiddetto gruppo multicast al quale appartengono tutte le interfacce caratterizzate da quell'indirizzo multicast. Gli indirizzi multicast sono caratterizzati dal prefisso `0xff00::/8`.

Una tipologia di indirizzo del tutto nuova è l'indirizzo anycast IPv6 il quale identifica un insieme di interfacce che appartengono tipicamente a nodi (router) differenti. Un indirizzo anycast è indistinguibile nella forma un unicast global address, può essere assegnato solo ad un router e non deve essere utilizzato come indirizzo di sorgente per un pacchetto IPv6. Un pacchetto trasmesso ad un indirizzo anycast è inoltrato ad una delle interfacce identificate da quell'indirizzo (tipicamente quella più vicina).

A.3 Header ed estensioni

L'header IPv6 è di dimensione fissa pari a 40 byte, ed il formato è mostrato in Figura A.2.

Version è un campo di 4 bit contenente la versione (in questo caso 6) del protocollo. *Traffic Class* è un campo di 8 bit che identifica la tipologia di traffico (ad es. nella forma di Differentiated Service), e *Flow Label* è un campo su 20 bit utilizzabili per assegnare un etichetta per la Quality of Service (QoS, ad es. tramite Integrated Service). *Payload Length* è un campo su 16 bit che indica la lunghezza del pacchetto IPv6 escluso l'header; gli eventuali extension header sono compresi nella Payload Length. Il campo *Next Header* su 8 bit identifica il tipo di extension header (se presente) che segue l'header IPv6. Il campo *Hop Limit* su 8 bit è decrementato di 1 ad ogni inoltro del pacchetto: simile al campo TTL di

IPv4, quando Hop Limit scende a 0 il pacchetto viene scartato. Seguono i campi da 128 bit per l'indirizzo sorgente (*Source Address*) e destinatario (*Destination Address*).

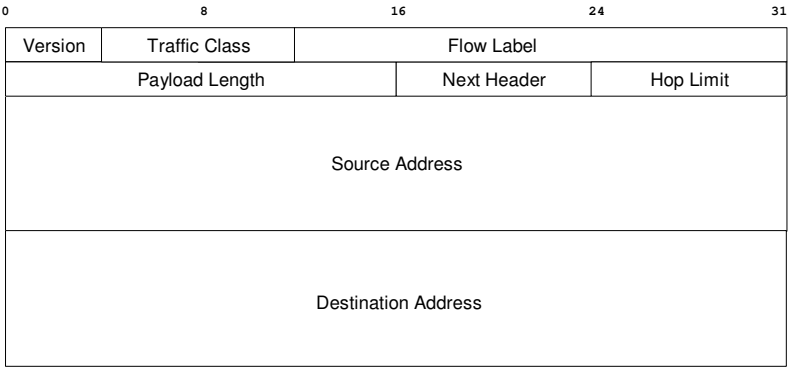

Figura A.2 – Formato dell'header IPv6

In IPv6 sono utilizzati i cosiddetti Extension Header per il supporto di opzioni quali source routing, cifratura, frammentazione, autenticazione e altre ancora. Gli extension header sono posti tra l'header ed il payload vero e proprio di un pacchetto IPv6 ed ognuno è caratterizzato da un codice presente nel campo Next Header dell'header che precede. Esemplifichiamo l'uso degli extension header con la Figura A.3: quando non sono presenti extension header, nel campo Next Header dell'header IPv6 è presente il codice del protocollo di livello superiore (in questo caso 6, associato al TCP) presente nel payload. Altrimenti è indicato il codice associato all'Option Header, fino ad arrivare all'ultimo extension header che conterra nel campo Next Header il codice del protocollo di livelo superiore presente nel payload.

Per le tipologie di extension header ed ulteriori dettagli si rimanda a [1].

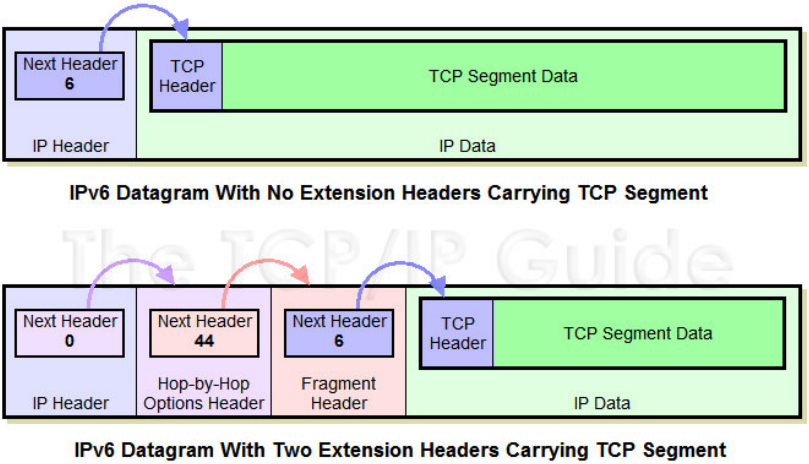

Figura A.3 – Esempio dell'uso di extension header[2]

A.4 Autoconfigurazione di rete

IPv6 fornisce agli host la capacità di autoconfigurare il prorio indirizzo, bypassando l'uso di un protocollo stateful di configurazione (tipo DHCP). Altri meccanismi (cosiddetti di Router Discovery) consentono ad un nodo di determinare automaticamente gli indirizzi dei router ed altri parametri di configurazione necessari per la connessione alla rete. Il processo di configurazione automatica si basa su messaggi broadcast inviati

[2] fonte: The TCP/IP Guide - http://www.tcpipguide.com/

periodicamente da un router sul proprio local-link denominati router advertisement (RA). Questi messaggi di tipo ICMPv6 contengono informazioni relative alla sottorete locale quali il prefisso di rete e l'indirizzo del default router. Se un nodo desidera autoconfigurarsi su questo link preleva il relativo prefisso dal messaggio RA e vi appende il proprio interface ID (MAC Address). Il nuovo indirizzo così ottenuto dovrà sottoporsi alla procedura DAD (Duplicate Access Detection) alla quale acceniamo nel prossimo paragrafo, per assicurarsi che non ci siano altre interfacce con quell'indirizzo sul link, prima di essere definitivamente assegnato all'interfaccia.

A.5 Duplicate Address Detection (DAD)

La procedura di Duplicate Address Detection (DAD) sfrutta i messaggi DHCPv6 [5] di Neighbor Solicitation e Neighbor Advertisement per verificare l'univocità degli indirizzi autoconfigurati da un nodo [36]. In particolare il nodo invia un messaggio broadcast di Neighbor Solicitation per chiedere a tutti i nodi della rete se l'indirizzo che ha creato automaticamente è già associato a qualche interfaccia. Questo è realizzato ponendo l'indirizzo :: (unspecified-IPv6 address) come indirizzo sorgente e l'indirizzo autoconfigurato come Target Address della Solicitation.

Se il mittente non riceve alcuna risposta di Neighbor Advertisement entro un timeout fissato, allora il nodo assume che nessun nodo stia utilizzando l'indirizzo autoconfigurato e lo assegna alla propria interfaccia.

A.6 Sicurezza ed IPsec

IPv6 (come pure IPv4) non include nessun meccanismo di sicurezza ed è pertanto passibile di molteplici tipi di attacchi. Ad esempio, si parla di packet sniffing quando i pacchetti in transito vengono letti da un nodo posto tra mittente e destinatario, il quale riesce ad acquisire così informazioni riservate (come ad esempio una password oppure un numero di carta di credito). Altri tipi di attacchi sono noti come IP spoofing e connection hijacking. Nel primo caso si tratta di una falsificazione dell'indirizzo mittente, nell'altro caso il sabotatore si inserisce in una comunicazione in corso, introducendo dati errati.

"La" risposta alle richieste di sicurezza per IPv6 è lo standard IPsec, che prevede la cifratura e l'autenticazione dei pacchetti IPv6 a livello di rete. I meccanismi di sicurezza risultano così trasparenti ai protocolli e alle applicazioni di livello superiore.

IPsec prevede l'uso di due differenti extension header, AH (Authentication Header) ed ESP (Encapsulated Security Payload), per fornire l'autenticazione, l'integrità e la confidenzialità della comunicazione.

Ci sono due modelli di funzionamento detti rispettivamente tunnel mode e transport mode: nel tunnel mode il datagramma IP viene completamente incapsulato (e autenticato/cifrato) in un nuovo datagramma IP., mentre in transport mode solo il payload del datagramma IP viene processato da IPsec che inserisce il proprio header tra l'header IP ed i livelli superiori (si veda Figura A.4).

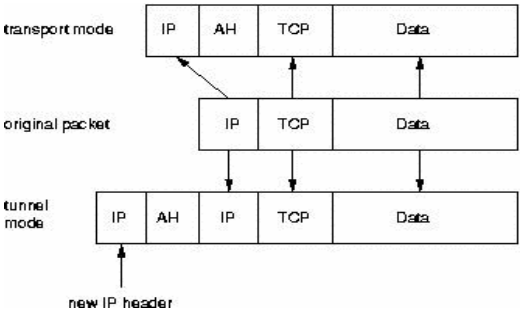

Figura A.4 – Esempio di tunnel mode e transport mode

Per maggiori dettagli si rimanda a [37], che è anche la fonte per la Figura A.4.

Appendice B
Messaggi e opzioni

B.1 Binding Update

Questo messaggio è identificato dal valore 5 nel campo MH Type ed è utilizzato da un nodo mobile per notificare ad altri nodi (CN, HA) un nuovo Care-of Address acquisito. Il formato del messaggio BU è mostrato in Figura B.1.

Il campo Sequence Number è utilizzato dal mittente per abbinare una BA ricevuta ad una BU precedentemente inviata e dal ricevente per ordinare le richieste provenienti da un nodo mobile.

I flag sono utilizzati con queste specifiche:

- Acknowledge (A): settato dal MN per richiedere l'invio di una BA alla ricezione della BU;
- Home Registration (H): settato dal MN per richiedere che il router destinatario della BU (che deve avere lo stesso prefisso dell'Home Address) sia l'Home Agent per il MN;

- Link-local Address Compatibility (L): settato quando l'Home Address riportato dal nodo mobile ha lo stesso identificatore di interfaccia dell'indirizzo link-local del MN;
- Key Management Mobility Capability (K): questo bit è processato solo dagli Home Agent; quando è settato a 0 indica che il protocollo usato per stabilire la IPSEC SA tra MN e HA non sopravvive agli spostamenti, oppure che la IPSEC SA è stata configurata manualmente.

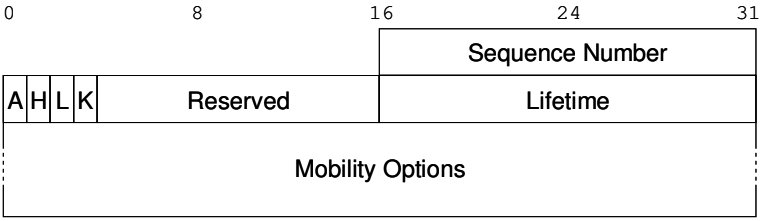

Figura B.1 – Messaggio di Binding Update

Il campo Reserved è riservato ad usi futuri, mentre il campo Lifetime contiene il numero di unità di tempo restanti prima che la corrispondenza HoA-CoA debba considerarsi esaurita; l'unità di tempo è pari a 4 secondi, ed un valore pari a 0 significa che la entry nella Binding Cache del ricevente va eliminata.

Un messaggio di Binding Update può contenere anche opzioni del tipo Type-Length-Value (TLV): in assenza di opzioni, è necessario includere nel campo Mobility Options padding pari a 4 ottetti. Le opzioni ammesse sono Binding Authorization Data, Nonce Indices (obbligatorie per BU inviate a CN), Alternate Care-of Address.

B.2 Binding Acknowledgement

Questo messaggio è identificato dal valore 6 nel campo MH Type ed è utilizzato da CN ed HA per riscontrare una BU ricevuta. Il formato del messaggio BA è mostrato in Figura B.2.

Il campo Status è molto importante poiché contiene l'esito della richiesta di Binding Update. Valori entro il numero decimale 128 indicano un successo della richiesta:

- 0 = BU accettata;
- 1 = BU accettata ma scoperta prefisso necessaria;

mentre valori uguali o superiori a 128 indicano un rifiuto della richiesta, tra i quali citiamo:

- 128 = motivo non specificato;
- 129 = rifiuto di tipo amministrativo;
- 130 = risorse insufficienti.

Il campo Lifetime e il flag K hanno significati identici a quanto detto per le BU, con l'aggiunta che un Correspondent Node non deve mai settare il flag K a 1. Il campo Sequence Number, copiato dalla BU ricevuta, consente la corrispondenza tra BU e BA.

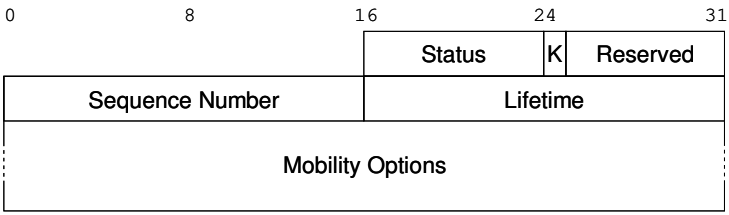

Figura B.2 - Messaggio di Binding Acknowledgement

Un messaggio di Binding Acknowledgement può contenere anche opzioni del tipo Type-Length-Value (TLV): in assenza di opzioni, è necessario includere nel campo Mobility Options padding pari a 4 ottetti. Le opzioni ammesse sono Binding Authorization Data (obbligatoria per BA inviate da CN), Binding Refresh Advice.

B.3 Binding Error

Questo messaggio è identificato dal valore 7 nel campo MH Type ed è utilizzato da un Correspondent Node per segnalare un errore relativo alla mobilità, come un inappropriato tentativo di usare una Home Address Destination option senza l'esistenza di una binding entry. Il formato del messaggio BA è mostrato in Figura B.3.

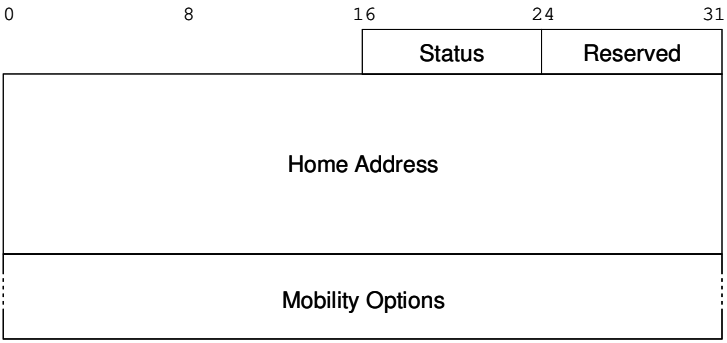

Figura B.3 - Messaggio di Binding Error

Il campo Status indica il motivo del messaggio, il campo Home Address contiene l'indirizzo che ha generato il messaggio d'errore, la

definizione di opzioni per questo tipo di messaggio è lasciata a future estensioni. Se non ci sono opzioni non è necessario alcun padding aggiuntivo e il campo Header Length del Mobility Header contiene il valore decimale 2.

B.4 Type 2 Routing Header

Il routing header di tipo 2 è definito in Mobile IPv6 per permettere ad un pacchetto inviato da un Correspondent Node di raggiungere il Care-of Address di un Mobile Node. All'interno del routing header di tipo 2 è presente l'Home Address del MN.

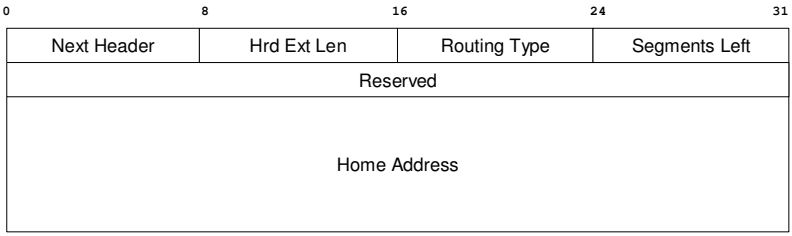

Figura B.4 – Formato del Type 2 Routing Header

Il campo Next Header identifica il tipo di header che segue il Mobility Header, utilizzando gli stessi valori del campo IPv6 Next Header. Il campo Hdr Ext Len è posto a 2 indicando la dimensione del routing header a gruppi di 8 ottetti (il primo gruppo non è compreso). Il campo Routing Type è posto a 2 indicando il tipo di routing header. Il campo Segments Left è posto ad 1 ad indicare il numero di nodi intermedi prima della destinazione finale: infatti dopo aver raggiunto il Care-of Adress del

MN manca un nodo, ovvero l'Home Address, alla consegna del pacchetto. Infine nel campo Home Address su 128 bit è inserito l'Home Address del nodo mobile destinatario del pacchetto.

B.5 Home Address Option

L'opzione Home Address è utilizzata da un nodo mobile nei pacchetti inviati quando esso si trova in una Foreign Network (ad esempio nelle BU) per informare il destinatario sul proprio Home Address. Questa opzione è trasportata dall'extension header Destination Option di IPv6 [1] ed è annunciata dal valore 60 del campo Next Header.

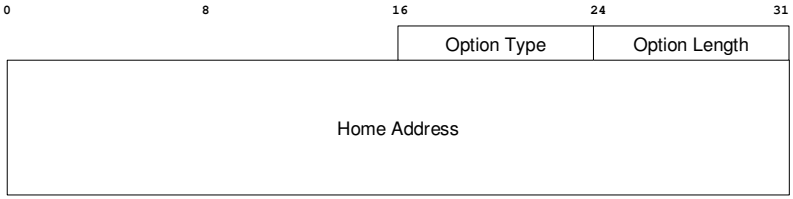

Figura B.5 – Formato dell'opzione Home Address

Il campo *Option Type* contiene l'identificatore dell'opzione, ovvero il valore decimale 201, mentre il campo *Option Length* è sempre posto al valore decimale 16, poiché indica la lunghezza dell'opzione a multipli di 8 bit esclusi i campi Option Type ed Option Length. Il campo *Home Address* su 128 bit contiene, ovviamente, l'Home Address del nodo mobile.

B.6 Mobility Message Authentication Option

Esaminiamo il formato della Mobility Message Authentication Option mostrato in Figura B.6:

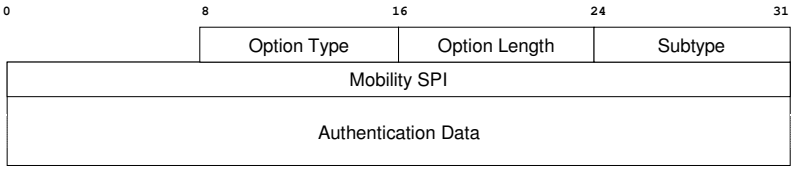

Figura B.6 – Formato della Mobility Message Authentication Option

Il campo *Option Type* su 8 bit identifica il tipo di Mobility Option, che per questo tipo di opzione è stato fissato da IANA nel valore decimale 9. Il campo *Option Length* è di tipo intero senza segno su 8 bit e rappresenta in byte la somma delle lunghezze dei campi Subtype, Mobility SPI e Authentication Data.

Il campo *Subtype* su 8 bit contiene un numero che identifica l'entità o il meccanismo che autentica il messaggio. Il valore 1 è associato al MN-HA Mobility Message Authentication Option, mentre il valore 2 è associato al MN-AAA Mobility Message Authentication Option.

Il campo *Mobility SPI*, dove SPI è l'acronimo di Security Parameter Index, su 32 bit contiene un valore compreso tra 256 e 4294967295 (in quanto i valori da 0 a 255 sono riservati) utilizzato per identificare una shared-key-based mobility security association in uso tra due entità.

Appendice C
Testing

C.1 Open Diameter

Una volta completato il setup di Open Diameter su una macchina Linux abbiamo effettuato due prove per testarne il corretto funzionamento: la prima con il metodo di autenticazione EAP-TLS e la seconda con MD5.

C.1.1 Test con EAP-TLS

Per testare le funzionalità di EAP-TLS abbiamo utilizzato degli script forniti da Pedro Garcia Segura: questa volta i comandi da lanciare nelle due finestre sono `./server_test_tls` e `./client_test_tls` (Figura C.1).

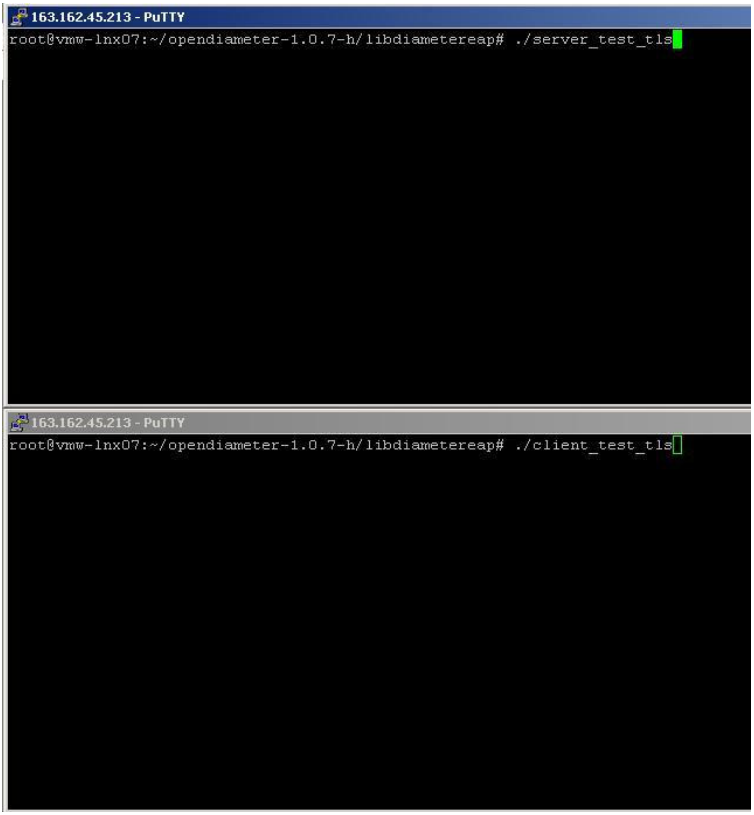

Figura C.1 – Shell server e client per test EAP-TLS

Questa volta, terminato lo scambio di messaggi CER/CEA, digitando 1 e 'Invio' nella shell del client inizia l'autenticazione via EAP-TLS, e compare la richiesta di username nella shell del client (Figura C.2). Inserendo l'username di test "ohba" e premendo 'Invio' inizia la procedura di autenticazione EAP che si conclude con successo (Figura C.3).

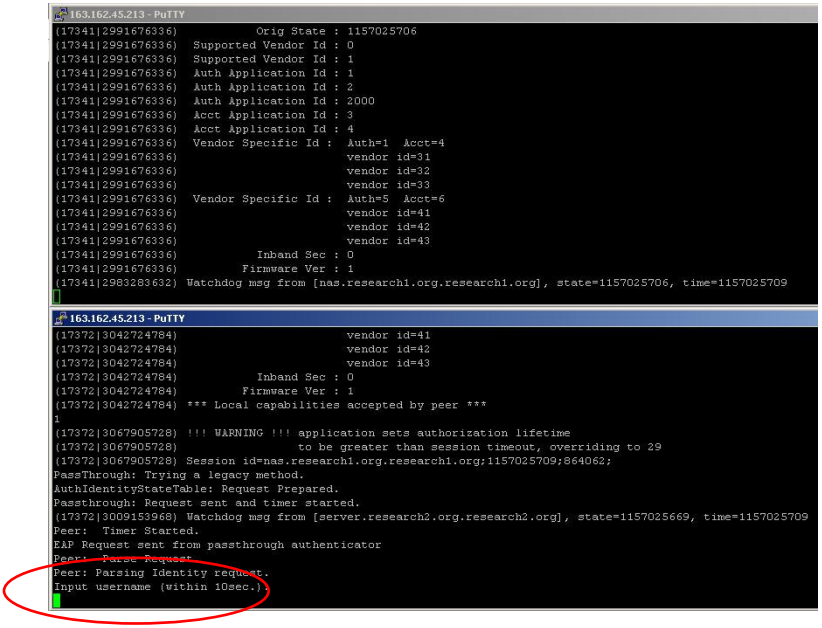

Figura C.2 - Richiesta username client

C.1.2 Test con MD5

Per MD5 inseriamo i nomi degli host manualmente nel file `/etc/hosts`:

```
2006::1 nas.research1.org
2006::2 server.research2.org
```

e configuriamo entrambi gli indirizzi sull'interfaccia di rete della nostra macchina:

```
ip addr add 2006::1 dev eth1
ip addr add 2006::2 dev eth1
```

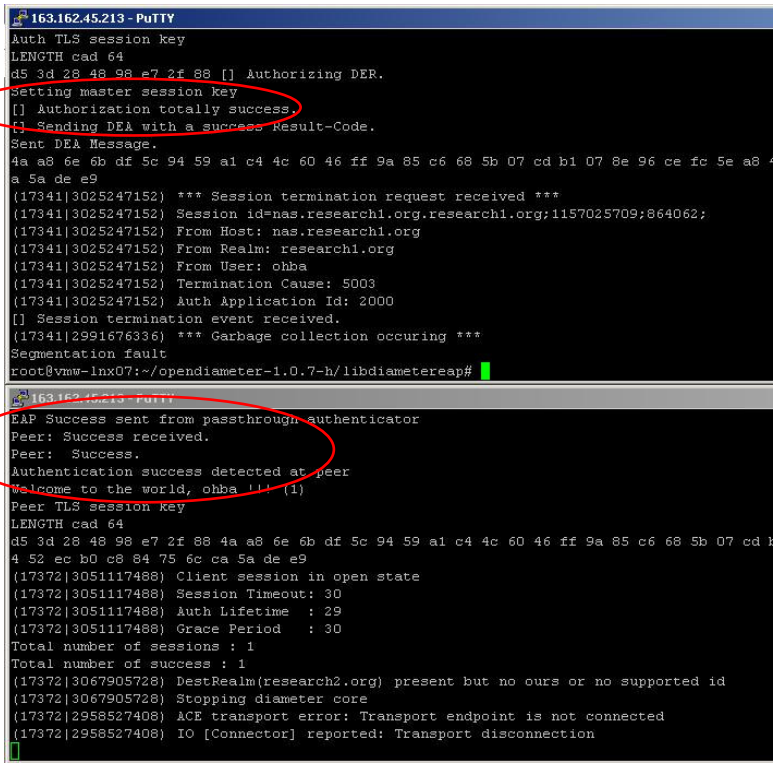

Figura C.3 – Autenticazione EAP-TLS avvenuta con successo

Una volta controllata la raggiungibilità dei due indirizzi abbiamo aperto due shell, una per simulare un server e l'altra per un client (vedi Figura C.4).

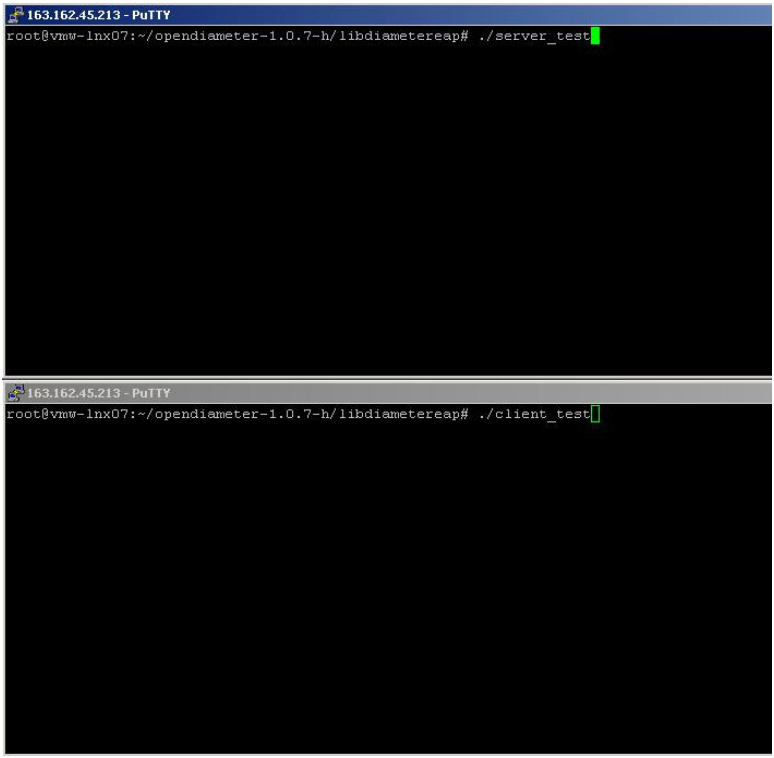

Figura C.4 – Shell server e client per test MD5

Dalla directory `../opendiameter-1.0.7-h/libdiametereap` abbiamo lanciato i comandi `./server_test` e `./client_test`: una volta instaurata la sessione Diameter e completato lo scambio di messaggi CER/CEA (Figura C.5) digitando 1 e 'Invio' nella shell del client inizia l'autenticazione via MD5, che andando a buon fine sancisce la riuscita del test (Figura C.6).

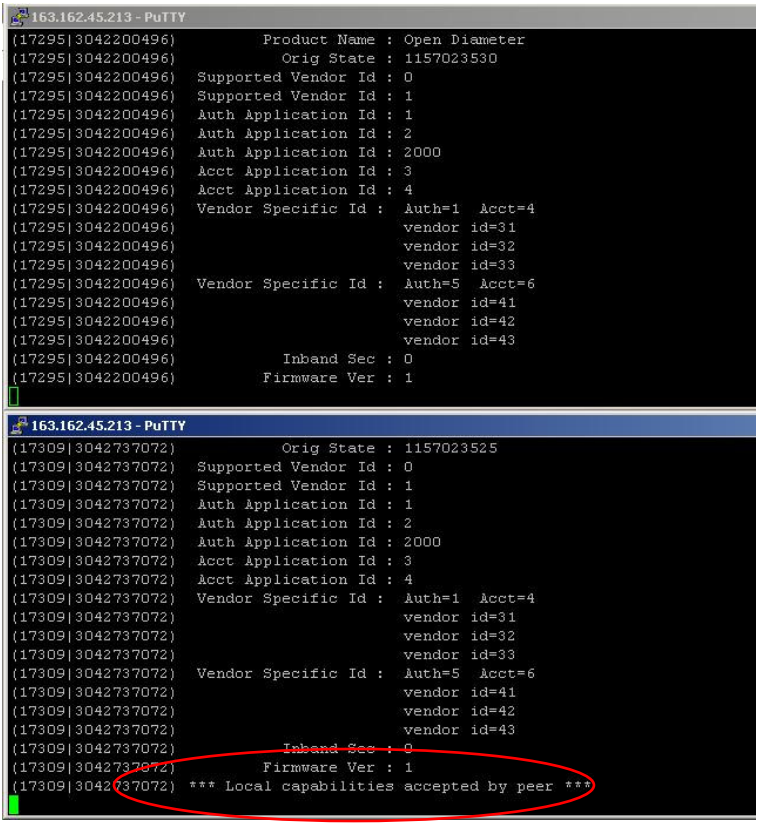

Figura C.5 – Fine scambio messaggi CER/CEA

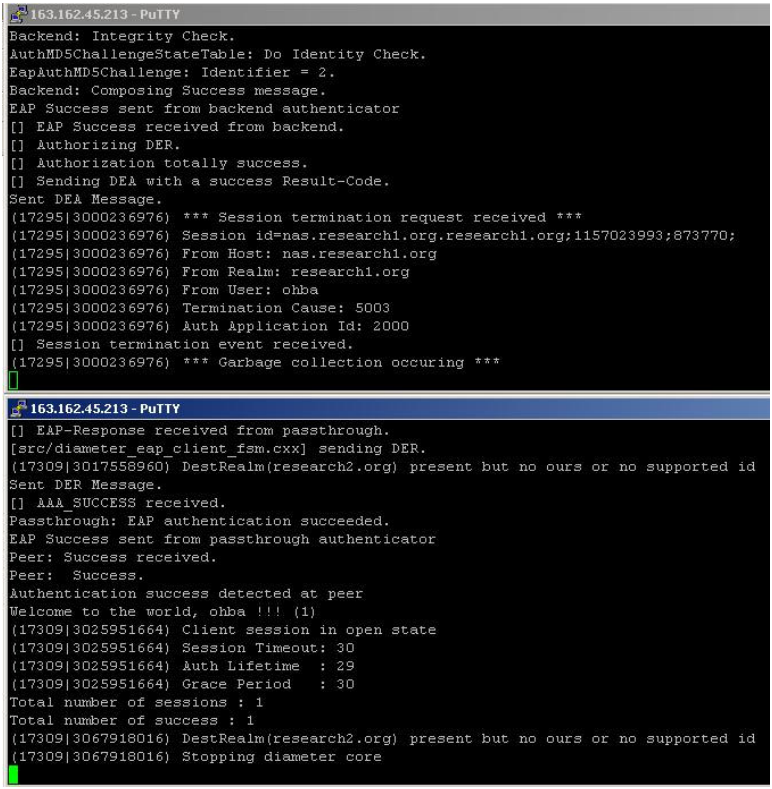

Figura C.6 – Autenticazione EAP-MD5 avvenuta con successo

Appendice D
Codice

D.1 MIPL

D.2 Diameter MIP6 Authorization application

D.2.1 Copyright

```
/* BEGIN_COPYRIGHT                                              */
/*                                                              */
/* Open Diameter: Open-source software for the Diameter and     */
/* Diameter related protocols                                   */
/*                                                              */
/* Copyright (C) 2002-2004 Open Diameter Project                */
/*                                                              */
/* This library is free software; you can redistribute it       */
/* and/or modify it under the terms of the GNU Lesser           */
/* General Public License as published by the Free Software     */
/* Foundation; either version 2.1 of the License, or (at        */
/* your option) any later version.                              */
/* This library is distributed in the hope that it will be      */
/* useful, but WITHOUT ANY WARRANTY; without even the           */
```

```
/* implied warranty of MERCHANTABILITY or FITNESS FOR A     */
/* PARTICULAR PURPOSE.  See the GNU Lesser General Public    */
/* License for more details.                                 */
/* You should have received a copy of the GNU Lesser General */
/* Public License along with this library; if not, write to  */
/* the Free Software Foundation, Inc., 59 Temple Place,      */
/* Suite 330, Boston, MA 02111-1307 USA.                     */
/*                                                           */
/* In addition, when you copy and redistribute some or the   */
/* entire part of the source code of this software with or   */
/* without modification, you MUST include this copyright     */
/* notice in each copy.                                      */
/* If you make any changes that are appeared to be useful,   */
/* please send sources that include the changed part to      */
/* diameter-developers@lists.sourceforge.net so that we can  */
/* reflect your changes to one unified version of this       */
/* software.                                                 */
/* END_COPYRIGHT                                             */
```

D.2.2 test/client_test.cxx

```
/*
client_test.cxx
Client test program for Diameter MIP6 Application
Written by Luca Battistoni
*/

/*
------------------------------------------------------------
Brief explanation on what is done in this sample program.
------------------------------------------------------------
This program opens a listening socket towards MIPL on port
47901. Every request coming from MIPL creates a new
authorization session with user NAI got from MIPL.
*/

#include <iostream>
#include <ace/Log_Msg.h>
#include <ace/OS.h>
#include <ace/Atomic_Op_T.h>
#include "diameter_api.h"
#include "diameter_mip6_client_session.hxx"

typedef AAA_JobHandle<AAA_GroupedJob> MyJobHandle;

class ClientData;
```

```
class HA_Application;

/// Task class used in this sample program.
class Mip6Task : public AAA_Task
{
public:
   /// Constructor.
   Mip6Task() : AAA_Task(AAA_SCHED_FIFO, "MIP6")
   {}

   /// Destructor.
   ~Mip6Task() {}
};

/// This class defines a transport.  When a message is sent
/// to a particular entity (e.g., an entity may be a client, a
/// HA or a MIP6 server depending on the role of the sender,
/// Transmit() method of the Channel object of the receiving
/// entity is called. Transmit() method can have sub-channels
/// which is used for distinguishing different types of
/// messages.
class Channel
{
public:
   Channel() {}
   virtual ~Channel() {}
   virtual void Transmit(DiameterMip6AuthorizationInfo
&authInfo)=0;
};


/******** Diameter MIP6 Client Session ********/

class MyDiameterMip6ClientSession : public
DiameterMip6ClientSession
{
public:
   MyDiameterMip6ClientSession(AAAApplicationCore& appCore,
MyJobHandle h)
   : DiameterMip6ClientSession(appCore, h)
   {}

   /// This virtual function is called when a MIP6 client
   /// session is aborted due to enqueue failure of a job
   /// or an event inside Diameter MIP6 client state
   /// machine.
   void Abort();

   /// Reimplemented from the parent class.
```

```
   void SignalSuccess();

   /// Reimplemented from the parent class.
   void SignalFailure();

   /// Reimplemented from the parent class.
   void SignalReauthorization();

   /// Reimplemented from the parent class.
   void SignalDisconnect() {}

   /// Reimplemented from the parent class.
   void SetDestinationRealm
   (AAA_ScholarAttribute<diameter_utf8string_t> &realm);

   /// Reimplemented from parent class.
   void SetAuthInfo(DiameterMip6AuthorizationInfo &authInfo);
};

class HA_Channel : public Channel
{
public:
   HA_Channel(MyDiameterMip6ClientSession& s) : session(s) {}

   void Transmit(DiameterMip6AuthorizationInfo &authInfo)
   { session.ForwardAuthorizationInfo(authInfo); }

   MyDiameterMip6ClientSession &session;
};

// My application session.
class HA_Application : public AAA_JobData
{

public:
   HA_Application(Mip6Task &task, AAAApplicationCore& appCore,
   ACE_Semaphore &sem, const char *u)
   : handle(MyJobHandle
   (AAA_GroupedJob::Create(appCore.GetTask().Job(), this,
"HA"))),
   diameter(boost::shared_ptr<MyDiameterMip6ClientSession>
   (new MyDiameterMip6ClientSession(appCore, handle))),
   semaphore(sem),
   rxChannel(HA_Channel(*diameter)),
   txChannel(0), user(u)
   {
      semaphore.acquire();
   }
   ~HA_Application() {}
```

```
    void Start(Channel *c)
    {
       txChannel = c;
       diameter->Start();
       USER_Info userInfo;
       userInfo.UserName() = user;
       c->Transmit(userInfo);
    }

    Channel* RxChannel() {return &rxChannel; }

    Channel& TxChannel() {return *txChannel; }

    MyDiameterMip6ClientSession &Diameter() {return *diameter; }

    ACE_Semaphore& Semaphore() {return semaphore; }

private:
    MyJobHandle handle;
    boost::shared_ptr<MyDiameterMip6ClientSession> diameter;
    ACE_Semaphore &semaphore;
    HA_Channel rxChannel;
    Channel *txChannel;
    const char *user;
};

// ----------------- Definition --------------
void
MyDiameterMip6ClientSession::Abort()
{
    std::cout << "Diameter MIP6 client session aborted." <<
std::endl;
    DiameterMip6ClientSession::Stop();
    JobData(Type2Type<HA_Application>()).Semaphore().release();
}

void
MyDiameterMip6ClientSession::SignalSuccess()
{
    AAA_LOG(LM_DEBUG, "Client authorization successful.\n");
    JobData(Type2Type<HA_Application>()).Semaphore().release();
}

void
MyDiameterMip6ClientSession::SignalFailure()
{
    AAA_LOG(LM_DEBUG, "Client authorization failed.\n");
    Abort();
```

```
}

void
MyDiameterMip6ClientSession::SignalReauthorization()
{
   AAA_LOG(LM_DEBUG, "Client Re-authorization triggered (to be
implemented).\n");
   Abort();
}

void
MyDiameterMip6ClientSession::SetDestinationRealm
(AAA_ScholarAttribute<diameter_utf8string_t> &realm)
{
   std::string& userName = AuthorizationInfo().UserName();
   size_t pos = userName.find('@');
   if (pos != std::string::npos) {
      pos ++;
      realm.Set(std::string(userName, pos, userName.length() -
pos));
   }
   else {
      Abort();
   }
}

class MyInitializer
{
public:
   MyInitializer(Mip6Task &t, AAAApplicationCore &appCore)
   : task(t), applicationCore(appCore)
   {
      Start();
   }

   ~MyInitializer()
   {
      Stop();
   }

private:

   void Start()
   {
      InitTask();
      InitApplicationCore();
   }

   void Stop()
```

```
   {
      task.Stop();
   }

   void InitTask()
   {
      try {
         task.Start(10);
      }
      catch (...) {
         ACE_ERROR((LM_ERROR, "(%P|%t) Server: Cannot start
task\n"));
         exit(1);
      }
   }
   void InitApplicationCore()
   {
      ACE_DEBUG((LM_DEBUG, "[%N] Application starting\n"));
      if (applicationCore.Open("config/client.local.xml",
            task) != AAA_ERR_SUCCESS)
      {
         ACE_ERROR((LM_ERROR, "[%N] Can't open configuration
file."));
         exit(1);
      }
   }

   Mip6Task &task;

   AAAApplicationCore &applicationCore;
};

int
main(int argc, char *argv[])
{
   Mip6Task task;
   AAAApplicationCore applicationCore;
   MyInitializer initializer(task, applicationCore);

   int num = 1;

   ACE_Semaphore semaphore(num);

   std::auto_ptr<HA_Application> haApp;

   char *ip;                                                  /* ip
address of the servers */
   ip = "127.0.0.1";
   int bytercv;                                               /*
```

```
number of received char */
   struct socket_message req;                                    /*
request and reply messages */

   /* Socket toward DIAMETER (out) */
   int sck_DIAMETER;                                             /*
socket descriptor */
   unsigned int port_DIAMETER = 47901;                           /*
server port */
   struct sockaddr_in address_DIAMETER;                          /*
structure of the address/port descriptor */
   struct sockaddr_in client_DIAMETER;                           /*
structure that describe the remote cl*/
   unsigned int client_addr_len_DIAMETER;

   /* Address preparation DIAMETER */
   init_sock_addr(&address_DIAMETER, ip, port_DIAMETER);
   client_DIAMETER.sin_family = AF_INET;
   client_addr_len_DIAMETER = sizeof(client_DIAMETER);

   /* Socket creation DIAMETER*/
   printf("Input toward DIAMETER ");
   socket_init(&sck_DIAMETER);

   /* Port and address association DIAMETER*/
   printf("Input toward DIAMETER ");
   socket_bind(sck_DIAMETER, address_DIAMETER);

   /* Interface toward MIPL */
   printf("*************************************\n");
   printf("*   Waiting for username from MIPL   *\n");
   printf("*************************************\n");

   /* MIPL sends to DIAMETER a user identity to be authorized
*/
   while(1){
      bytercv = recvfrom(sck_DIAMETER, &req, sizeof(req), 0,
(struct sockaddr *) &client_DIAMETER,
&client_addr_len_DIAMETER);
      if( bytercv == -1 )
         ACE_ERROR((LM_ERROR, "[%N] Error on the data
reception.\n"));
      printf("*** Message type received from MIPL = %d
***\n",req.type);

      /* Now read the MN_ID_NAI of the message and send back
result */
      if(req.type == AUTHORIZATION_REQUEST) {
         printf("*** Checking authorization for user: %s
```

```
***\n",req.MN_ID_NAI);
          haApp = std::auto_ptr<HA_Application>
                  (new HA_Application(task, applicationCore,
semaphore, req.MN_ID_NAI));
          haApp->Start(haApp->RxChannel());
       }
   }

   return 0;
}
```

D.2.3 test/server_test.cxx

```
/*
   server_test.cxx
   Server test program for Diameter MIP6 Application
   Written by Luca Battistoni
*/

#include <iostream>
#include <ace/Log_Msg.h>
#include <ace/OS.h>
#include "diameter_api.h"
#include "diameter_mip6_parser.hxx"
#include "diameter_mip6_server_session.hxx"
#include "diameter_mip6_authinfo.hxx"

typedef AAA_JobHandle<AAA_GroupedJob> MyJobHandle;

/******** Diameter MIP6 Server Session ********/

class BackendSession : AAA_Job
{
 public:

  BackendSession(MyJobHandle &handle) : handle(handle) {}

  void Receive(DiameterMip6AuthorizationInfo &authInfo)
  {
    bool authSuccess;
    AAA_LOG(LM_DEBUG, "Username = %s.\n",
authInfo.UserName().c_str());
    diameter_utf8string_t user = authInfo.UserName().c_str();
    if (authInfo.AuthorizationType() ==
MIP6_AUTHORIZATION_TYPE_USER)
      authSuccess = ((USER_Info&)authInfo).Validate(user);
    else
```

```
      authSuccess = false;
    if (authSuccess)
      Success();
    else
      Failure();
  }

  int Serve() { return 0; }

  int Schedule(AAA_Job*, size_t) { return 0; }

  void Success();

  void Failure();

 private:
  std::string username;
  std::string password;
  MyJobHandle& handle;
};

class MyDiameterMip6ServerSession : public
DiameterMip6ServerSession
{
 public:
  MyDiameterMip6ServerSession(AAAApplicationCore& appCore,
              diameter_unsigned32_t appId=Mip6ApplicationId)
    : DiameterMip6ServerSession(appCore, appId),
           handle(MyJobHandle(AAA_GroupedJob::Create
             (appCore.GetTask().Job(), this, "backend"))),
           session(boost::shared_ptr<BackendSession>(new
              BackendSession(handle)))
  {
    this->Start();
  }

  void Start() throw(AAA_Error)
  {
    DiameterMip6ServerSession::Start();
  }

  /// This virtual function is called when a MIP6 server
  /// session is aborted due to enqueue failure of a job or an
  /// event inside Diameter MIP6 server state machine.
  void Abort()
  {
    std::cout << "Diameter MIP6 server session aborted." <<
       std::endl;
    DiameterMip6ServerSession::Stop();
```

```
  }

  BackendSession& Session() { return *session; }

  /// This virtual function is called when an authentication
  /// information is passed to the backend.
  void ForwardAuthorizationInfo(DiameterMip6AuthorizationInfo
&authInfo);

 private:
  MyJobHandle handle;
  boost::shared_ptr<BackendSession> session;
};

// ----------------- Definition --------------
void BackendSession::Success()
  {
    std::cout << "Success sent from backend" << std::endl;
    handle.Job().Data(Type2Type<MyDiameterMip6ServerSession>())
      ->SignalSuccess();
  }
void BackendSession::Failure()
  {
    std::cout << "Failure" << std::endl;
   handle.Job().Data(Type2Type<MyDiameterMip6ServerSession>())
     ->SignalFailure();
  }

void
MyDiameterMip6ServerSession::ForwardAuthorizationInfo
(DiameterMip6AuthorizationInfo &authInfo)
{
  std::cout << "Auth-Info forwarded to backend" << std::endl;
  Session().Receive(authInfo);
}

typedef
AAAServerSessionClassFactory<MyDiameterMip6ServerSession>
MyServerFactory;

class MyInitializer
{
 public:
  MyInitializer(AAA_Task &t, AAAApplicationCore &appCore)
    : task(t), applicationCore(appCore)
  {
    Start();
  }
```

```
  ~MyInitializer()
  {
    Stop();
  }

 private:

  void Start()
  {
    InitTask();
    InitApplicationCore();
    myAuthFactory = std::auto_ptr<MyServerFactory>
      (new MyServerFactory(applicationCore,
          Mip6ApplicationId));
    applicationCore.RegisterServerSessionFactory
      (myAuthFactory.get());
  }

  void Stop() {}

  void InitTask()
  {
     try {
        task.Start(10);
     }
     catch (...) {
        ACE_ERROR((LM_ERROR, "(%P|%t) Server: Cannot start
          task\n"));
        exit(1);
     }
  }
  void InitApplicationCore()
  {
    AAA_LOG(LM_DEBUG, "[%N] Application starting\n");
    if (applicationCore.Open("config/server.local.xml",
                             task) != AAA_ERR_SUCCESS)
      {
   AAA_LOG(LM_ERROR, "[%N] Can't open configuration file.");
    exit(1);
      }
  }

  AAA_Task &task;
  AAAApplicationCore &applicationCore;
  std::auto_ptr<MyServerFactory> myAuthFactory;
};

int
main(int argc, char *argv[])
```

```
{
  AAA_Task myTask;
  AAAApplicationCore applicationCore;
  MyInitializer initializer(myTask, applicationCore);

  while (1)
      ACE_OS::sleep(1);
  return 0;
}
```

D.2.4 include/diameter_mip6_authzinfo.hxx

```
#ifndef __MIP6_AUTHINFO_H__
#define __MIP6_AUTHINFO_H__

#include "framework.h"
#include "diameter_mip6_parser.hxx"
#include "aaa_user_db.h"

enum DiameterMip6AuthorizationType {
  MIP6_AUTHORIZATION_TYPE_NONE,
  MIP6_AUTHORIZATION_TYPE_USER,
};

/// The base class for Diameter MIP6 authentication information
class DiameterMip6AuthorizationInfo
{
public:
  DiameterMip6AuthorizationInfo(DiameterMip6AuthorizationType
t=MIP6_AUTHORIZATION_TYPE_NONE):authenticationType(t),
prompt(false)
{}

  DiameterMip6AuthorizationInfo (diameter_utf8string_t&
username, DiameterMip6AuthorizationType
t=MIP6_AUTHORIZATION_TYPE_NONE): authenticationType(t),
userName(username), prompt(false)
  {}

  DiameterMip6AuthorizationType& AuthorizationType()
  { return authorizationType; }

// Used for setting/getting username.
  diameter_utf8string_t& UserName() { return userName; }

private:
  DiameterMip6AuthorizationType authenticationType;
```

```
  diameter_utf8string_t userName;
  bool prompt;
};

/// The class for storing user information.
class USER_Info : public DiameterMip6AuthorizationInfo
{
public:
  USER_Info(diameter_utf8string_t& username):
DiameterMip6AuthorizationInfo(username,MIP6_AUTHORIZATION_TYPE_
USER), User(username), m_UserEntry()
{}

  USER_Info() :
    DiameterMip6AuthorizationInfo(MIP6_AUTHORIZATION_TYPE_USER)
  {}
  // Used for setting/getting username.
  diameter_utf8string_t& User() { return User; }
  // ****add for lifetime***
  diameter_unsigned32_t& Lifetime() { return lifetime; }

  bool Validate(diameter_utf8string_t& user)
  {
     AAA_UserDbLoader userDbLoader("config/aaa_user_db.xml");
     AAA_LOG(LM_INFO, "**** XML user database loaded ****\n");
     AAA_USER_DB().Dump();
     m_UserEntry = (AAA_UserEntry*)AAA_USER_DB().Match(user);
     if (m_UserEntry == NULL) {
         AAA_LOG(LM_INFO,"(%P|%t) ERROR !!! No matching entry
in user db \n");
         return false;
     }
     else {
    lifetime = m_UserEntry->Lifetime();
    return true;
     }
   }

private:
  diameter_utf8string_t User;
  AAA_UserEntry *m_UserEntry;
  diameter_unsigned32_t lifetime;
};
```

Bibliografia

[1] Deering, S., Hinden, R., “Internet protocol, Version 6 (IPv6) Specification”, RFC 2460, dicembre 1998

[2] Sito web http://www.ietf.org/

[3] Deering, S., Hinden, R., “Internet Protocol Version 6 (IPv6) Addressing Architecture”, RFC 3513, aprile 2003

[4] Johnson, D. et al., “Mobility Support in IPv6”, RFC 3775, giugno 2004

[5] Droms, R. et al., “Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, luglio 2003

[6] Kaufman, C., “Internet Key Exchange (IKEv2) Protocol”, RFC 4306, dicembre 2005

[7] Patel, A. et al., “Authentication Protocol for Mobile IPv6”, RFC 4285, gennaio 2006

[8] Patel, A., Giaretta, G., “Problem Statement for Bootstrapping Mobile IPv6 (MIPv6)”, RFC 4640, settembre 2006

[9] Giaretta, G. et al., “Mobile IPv6 bootstrapping in split scenario”,

draft-ietf-mip6-bootstrapping-split-03 (work in progress), ottobre 2006

[10] Chowdhury, K., Yegin, A., “MIP6-bootstrapping via DHCPv6 for the Integrated Scenario”, draft-ietf-mip6-bootstrapping-integrated-02 (work in progress), febbraio 2007

[11] Jang, H. et al., "DHCP Option for Home Agent Discovery in MIPv6", draft-ietf-mip6-hiopt-02 (work in progress), febbraio 2007

[12] Devarapalli, V., Dupont, F., “Mobile IPv6 Operation with IKEv2 and the revised IPsec Architecture”, draft-ietf-mip6-ikev2-ipsec-08 (work in progress), dicembre 2006

[13] Giaretta, G. et al., “AAA Goals for Mobile IPv6”, draft-ietf-mip6-aaa-ha-goals-03 (work in progress), settembre 2006

[14] Rigney, C. et al., “Remote Authentication Dial In User Service (RADIUS)”, RFC 2865, giugno 2000

[15] Aboba, B. et al., “Criteria for Evaluating AAA Protocols for Network Access", RFC 2989, novembre 2000

[16] Calhoun, P. et al., “Diameter Base Protocol”, RFC 3588, settembre 2003

[17] Sito web http://www.tschofenig.com/twiki/bin/view/Dime/WebHome

[18] Fajardo, V., Loughney, J., “Diameter Base Protocol”, draft-ietf-dime-rfc3588bis-01 (work in progress), gennaio 2007

[19] Calhoun, P. et al., “Diameter Network Access Server Application”, RFC 4005, agosto 2005

[20] Guttman, E. et al., “Service Location Protocol, Version 2”, RFC

2608, giugno 1999

[21] Crocker, D., Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, ottobre 2005

[22] Sito web http://www.iana.org/assignments/aaa-parameters

[23] Bournelle, J. et al., "Mobile IPv6 Bootstrapping using Diameter in the Split Scenario", draft-ietf-dime-mip6-split-00, giugno 2006

[24] Eronen, P. et al., "Diameter Extensible Authentication Protocol (EAP) Application", RFC 4072, agosto 2005

[25] Bournelle, J. et al., "Diameter Mobile IPv6: HA-to-AAAH support", draft-ietf-dime-mip6-split-01, ottobre 2006

[26] Patel, A. et al., "Authentication Protocol for Mobile IPv6", RFC 4285, gennaio 2006

[27] Haley, B. et al., "Mobility Header Home Agent Switch Message", draft-ietf-mip6-ha-switch-02 (work in progress), dicembre 2006

[28] Sito web http://www.mipl.mediapoli.com/

[29] Sito web http://www.opendiameter.org/

[30] Sito web http://www.opendiameter.org/twiki/bin/view/Diameter/OpenDiameterWiki

[31] Sito web http://www.cs.wustl.edu/~schmidt/ACE.html

[32] Sito web http://www.boost.org/

[33] Ohba, Y., "Framework API for Multithreading Task and Protocol State Machine", Alpha version, Documentation for Open Diameter (http://www.opendiameter.org), dicembre 2003

[34] Schmidt, D. C., "The ADAPTIVE Communications Environment, An Object-Oriented Netwrok Programming Toolkit for Developing

Communications Software", 11th Sun User Group conference, San Francisco (CA) - U.S.A, giugno 1993

[35] Sito web http://www.gnu.org/software/bison/

[36] Thomson, S, Narten, S., "IPv6 Stateless Address Autoconfiguration", RFC 2462, dicembre 1998

[37] Spenneberg, R., "IPsec HOWTO", sito web http://www2.autistici.org/ginox/ipsec-howto/ipsec-howto.html