ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
SEDE DI CESENA

SECONDA FACOLTÀ DI INGEGNERIA CON SEDE A CESENA

CORSO DI LAUREA SPECIALISTICA IN INGEGNERIA
ELETTRONICA E DELLE TELECOMUNICAZIONI

IMPLEMENTAZIONE DI RETI DI NUOVA GENERAZIONE (NGN)

Tesi in
PROGETTO DI RETI DI TELECOMUNICAZIONI LS

Relatore:
Prof. Ing.
FRANCO CALLEGATI

Correlatore:
Ing.
ALDO CAMPI

Presentata da:
FILIPPO ZANGHERI

Sessione II
Anno Accademico 2007–2008

A Elisa,
mia forza
mia debolezza
mia fonte di ispirazione

Indice

Elenco delle figure

Capitolo 1

Introduzione

Fin dagli albori delle comunicazioni di tipo elettrico/elettronico, il settore delle telecomunicazioni ha continuato a registrare andamenti di crescita di tipo esponenziale. Oggigiorno, il mondo dell'Information Technology (IT) avverte con sempre maggiore enfasi l'esigenza di gestire e integrare servizi e risorse di rete in uno scenario distribuito, sempre più dinamico e marcatamente eterogeneo. Il raggiungimento di questi obiettivi passa attraverso la rimozione di alcune barriere che normalmente separano i sistemi di elaborazione e i servizi applicativi dalle infrastrutture di rete.

L'interconnessione di sistemi e servizi IT o di Grid Computing è sempre più spesso affidata alle tecnologie di trasporto più promettenti nel campo dell'ottica, come ad esempio le cosiddette Intelligent Optical Networks. Grazie al progresso in questo settore, infatti, le reti ottiche possono verosimilmente far fronte ai requisiti di banda delle future applicazioni IT. Queste reti sono dotate di un piano di controllo, o Control Plane (CP), tipicamente basato su Generalized Multi-Protocol Label Switching (GMPLS)[1], il quale sostanzialmente automatizza il controllo della connettività di rete esponendo all'esterno le relative primitive, mediante apposite interfacce User-to-Network Interfaces (UNI). Tuttavia esso non è in grado di soddisfare richieste di servizio provenienti direttamente dalle applicazioni, a causa della diversità che esiste fra il modo di descrivere le risorse da parte delle applicazioni e quello correttamente interpretato dalle primitive del CP. Ciò si traduce nella necessità di nuovi elementi di rete che permettano un'interazione diretta fra applicazioni e la rete stessa.

Con “application-aware networking” ci si riferisce ad un tipo di architettura in grado di rendersi consapevole di quelli che sono i requisiti delle applicazioni che fanno uso della rete. Seguendo quanto detto sopra, lo scopo di questa tesi è di presentare una possibile soluzione architetturale per la realizzazione del concetto di application-aware networking, rifacendosi all'architettura di rete di nuova generazione (NGN) specificata dall'International Telecommunication Union (ITU-T)[2]. In particolare si intende fornire alle applicazioni dei validi strumenti che consentano loro di richiedere alla rete, in maniera diretta ed esplicita, le risorse applicative e di rete di cui intendono usufruire. Verrà mostrato come sia possibile raggiungere questo obiettivo attraverso l'accostamento originale di tecnologie e protocolli di rete affermati, per mezzo di un esempio implementativo dei blocchi funzionali su cui è basata l'architettura proposta.

Capitolo 2

Architettura proposta

L'idea fondamentale che sta alla base dell'architettura qui proposta è rendere l'infrastruttura di rete capace di assistere le applicazioni durante le fasi di pubblicazione, ricerca e riservazione delle risorse applicative e di rete. In particolare, l'idea è di permettere alle applicazioni di specificare in maniera diretta le proprie esigenze di tipo applicativo e di rete, per mezzo di descrizioni basate sui parametri percepiti dalle applicazioni stesse. Per raggiungere questo obiettivo, l'architettura necessita di:

- Dare alle applicazioni gli strumenti necessari per scambiare messaggi semanticamente ricchi con la rete, in modo da realizzare una negoziazione delle risorse di cui necessitano.
- Un linguaggio sufficientemente completo da poter essere usato per rappresentare i requisiti di connettività ad alto livello delle applicazioni, ossia in maniera del tutto indipendente dalla particolare tecnologia e topologia della rete di trasporto.
- Uno strumento che permetta di operare un controllo sulla rete indipendente dalla tecnologia, mappando le richieste di connettività ad alto livello su direttive comprensibili agli elementi di rete sottostanti.

Massimizzando l'uso di tecnologie già affermate, viene proposto l'uso del protocollo SIP (Session Initiation Protocol)[3] per la gestione della segnalazione fra gli utenti e l'uso del Resource Description Framework (RDF)[17] come base per la descrizione delle risorse di rete.

In questo capitolo verranno introdotti gli elementi di novità che costituiscono i blocchi fondamentali dell'architettura di application-aware networking su rete NGN proposta nel presente lavoro di tesi.

L'architettura logica è rappresentata in figura 1, nella quale sono messi in evidenza due strati. Lo strato di sessione è usato per la gestione della segnalazione delle applicazioni, mentre lo strato di trasporto rappresenta il Transport Stratum dell'architettura di rete NGN. per questo e altri dettagli associati all'architettura NGN si faccia riferimento al prossimo capitolo.

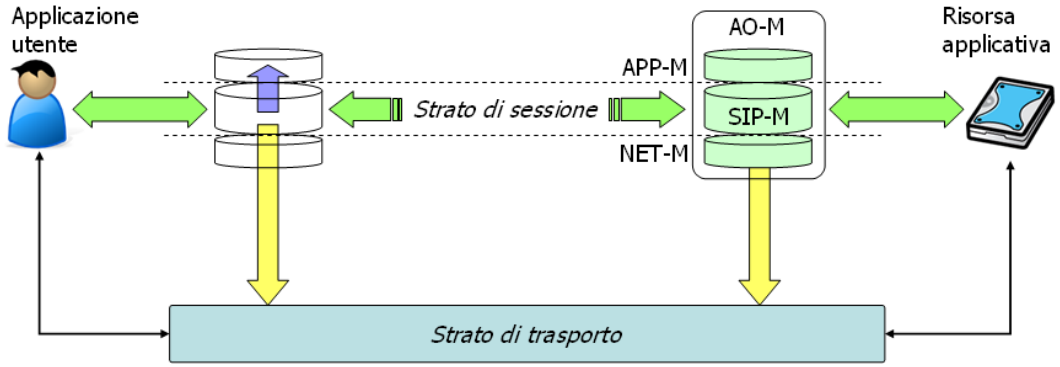

Figura 1. Architettura proposta

2.1 Application Oriented module

Nella descrizione dell'architettura presentata, il Service Stratum della rete NGN viene considerato come condensato nella sua Service Control Function (SCF), essendo quest'ultimo l'elemento di maggior interesse ai fini della caratterizzazione.

In questa architettura, la SCF è implementata da un modulo chiamato "Application Oriented Module (AO-M)", il quale è l'entità software che conferisce alla rete la capacità di essere "application-aware". Qui viene implementato lo strato di sessione, ed è a questo proposito che viene utilizzato il protocollo SIP. Il SIP infatti mette a disposizione tutte le primitive necessarie all'autenticazione e all'autorizzazione degli

utenti, alla creazione, gestione e terminazione di sessioni di qualunque genere, e alla localizzazione delle risorse in scenari distribuiti.

L'uso delle sessioni nella gestione delle richieste degli utenti permette di mantenere lo stato di tali richieste mappandolo sugli attributi delle sessioni associate.

Il modulo AO-M è logicamente posto sopra il Transport Stratum NGN e dialoga direttamente con le applicazioni. Esso è a sua volta suddiviso in tre moduli distinti:

1. Modulo SIP (SIP-M): è il modulo in cui risiedono le logiche SIP. Qui avviene la gestione delle sessioni di segnalazione fra gli utenti della rete. Questo è di fatto il blocco fondamentale del modulo AO-M ed incorpora le necessarie interfacce per interagire con i due moduli posti sopra e sotto di esso.

2. Modulo applicativo (APP-M): si occupa del parsing e dell'interpretazione delle descrizioni di risorse di tipo applicativo, incapsulate nei messaggi SIP.

3. Modulo di rete (NET-M): è incaricato di interagire con la rete sottostante, in particolare con le funzioni RACF del Transport Stratum NGN.

Combinando le funzionalità messe a disposizione da APP-M e SIP-M, la SCF è in grado di assistere le applicazioni nella pubblicazione, ricerca e riservazione delle risorse applicative. Questo può avvenire, ad esempio, attraverso l'uso di meccanismi SIP standard quali `PUBLISH`, `SUBSCRIBE` e `NOTIFY`, come indicato in [4].

Per mezzo del modulo NET-M, la SCF può interpretare le richieste di risorse di rete incapsulate nei messaggi SIP dalle applicazioni, e soddisfare i loro requisiti di connettività controllando la riservazione di tali risorse attraverso l'interazione con le Transport Control Functions.

Questa architettura è pensata per poter gestire sia richieste di risorse applicative che di rete. Tuttavia è al di fuori degli scopi di questa tesi descrivere l'aspetto riguardante la gestione delle risorse applicative; pertanto questo aspetto verrà posto in secondo piano.

2.2 Vocabolario RDF

L'idea di utilizzare un vocabolario RDF per descrivere informazioni relative alla rete non è nuova. Il Network Description Language (NDL)[5] è infatti utilizzabile per

la descrizione di informazioni legate alla topologia di una rete e agli elementi fisici che la costituiscono. Tuttavia, l'architettura che viene proposta deve far fronte a richieste di risorse di rete provenienti dalle applicazioni, pertanto occorre una sintassi differente e abbastanza generale da consentire la costruzione di documenti che rispecchino le esigenze di connetività delle applicazioni stesse. Occorre un vocabolario in grado di astrarre completamente rispetto agli aspetti topologici e tecnologici della rete, concentrandosi sulle caratteristiche di rete percepite dalle applicazioni.

Il vocabolario RDF introdotto in questa tesi è stato chiamato "Network Resource Description Language (NRDL)". Esso si propone come mezzo generale per la descrizione, quanto più completa possibile, delle risorse di rete. Per maggiori dettagli su questo vocabolario, si rimanda al capitolo in cui è discusso il lavoro svolto (cap. 5).

2.3 Principio di funzionamento

Con l'aiuto di un esempio, si procederà ora alla descrizione del funzionamento dell'architettura.

Si faccia riferimento allo scenario in cui un utente della rete NGN sia interessato ad un particolare servizio applicativo messo a disposizione da un application server, e che entrambe le entità siano state preventivamente autenticate ed autorizzate all'utilizzo della rete stessa.

2.3.1 Pubblicazione e scoperta delle risorse

I meccanismi con cui l'applicazione utente può venire a conoscenza dell'esistenza di un servizio applicativo possono essere implementati in vari modi attraverso l'uso di meccanismi standard SIP, come precedentemente sottolineato. Possono altresì essere messi in atto meccanismi analoghi per ciò che concerne la pubblicazione delle risorse di rete, ma ciò è al di là degli obiettivi di questa tesi.

Tramite i suddetti metodi l'applicazione utente è in grado di registrarsi presso un server SIP all'interno dello stesso dominio, per essere notificata riguardo la pubblicazione di determinate risorse applicative, oppure può farne esplicita richiesta.

L'applicazione utente si forma una visione della rete, ovvero ha un'astrazione ad alto livello di ciò che la rete rappresenta in termini di risorse (applicative e di rete), ma è completamente all'oscuro delle sue caratteristiche tecniche e topologiche.

2.3.2 Riservazione delle risorse

In base alle nozioni in suo possesso, l'applicazione utente crea una descrizione delle risorse applicative di cui intende fare uso durante la sessione di comunicazione con l'application server. Questa sessione può essere di qualunque tipo, da una sessione multimediale al trasferimento di dati su un sistema di archiviazione remoto. Per la descrizione delle risorse applicative possono essere usati protocolli come il Session Description Protocol (SDP)[10] o il Job Submission Description Language (JSDL), a seconda delle esigenze delle applicazioni stesse.

Oltre alla descrizione delle risorse di tipo applicativo, l'applicazione utente crea un opportuno documento NRDL recante la descrizione ad alto livello delle risorse di rete che intende utilizzare, ad esempio la descrizione delle connessioni richieste, con relativi parametri di qualità come bitrate, ritardo, jitter, ecc.

A questo punto viene composto un messaggio SIP di richiesta (INVITE), in formato multipart, nel quale vengono incapsulati i documenti di richiesta di risorse applicative e di rete. Questo messaggio viene trasmesso all'entità del Service Stratum NGN preposta alla segnalazione con gli utenti, ovvero la SCF, ed è destinato all'indirizzo SIP dell'application server, meglio noto come SIP Uniform Resource Identifier (URI).

La SCF, o meglio il modulo AO-M, riceve il messaggio di INVITE e ne estrae il contenuto NRDL. A questo punto la richiesta di risorse di rete viene comunicata, tramite il modulo NET-M, alle funzioni di controllo dello strato di trasporto NGN, ed in particolare alle RACF. Una volta stabilita l'ammissibilità della richiesta a livello di rete, le funzioni RACF procedono alla riservazione delle risorse di rete attraverso meccanismi dipendenti dalla tecnologia di trasporto usata. Per ogni risorsa di rete richiesta viene riportata una risposta che ne indica lo stato di riservazione (come minimo deve indicare se la riservazione è andata a buon fine oppure no). Il documento NRDL contenente le risorse di rete richieste e lo stato della loro riservazione viene reincapsulato nel messaggio SIP di richiesta originale e quindi inoltrato alla volta dell'application server, in questo modo si tiene traccia dello stato di riservazione delle risorse di rete

lungo il pecorso di segnalazione SIP e il server può prendere decisioni in base a tali informazioni.

A questo punto, l'application server verifica che la riservazione delle risorse di rete richieste dall'utente sia andata a buon fine e processa la richiesta di risorse applicative. Una volta eseguita la riservazione delle proprie risorse di tipo applicativo, il server trasmette un opportuno messaggio SIP di risposta, ad esempio "183 Session Progress" che contiene le proprie richieste di risorse di rete in formato NRDL ed eventualmente informazioni legate alla segnalazione applicativa.

Il modulo AO-M riceve la risposta del server, ne estrae il contenuto NRDL ed esegue gli stessi passi descritti sopra per la riservazione delle risorse di rete. Una volta giunte le risposte dallo strato di trasporto sullo stato di riservazione, il nuovo documento NRDL viene incapsulato nel messaggio di risposta e trasmesso all'applicazione utente.

A questo punto, se le riservazioni di risorse di rete e applicative sono andate a buon fine, la sessione di comunicazione si considera stabilita e può avvenire lo scambio di informazioni fra le due entità software attraverso la rete di trasporto NGN, con modalità che soddisfano i loro requisiti di connettività.

2.4 Punti di forza

L'elemento maggiormente vantaggioso dell'architettura presentata è dato dalla possibilità per l'applicazione di manifestare alla rete i propri requisiti di connettività in modo diretto e tramite l'uso di un linguaggio "user-oriented".

Un altro punto di forza è dato dallo sfruttamento del protocollo SIP nella segnalazione. Infatti la rete di segnalazione di tipo SIP offre un valido supporto nativo a scenari distribuiti e mette a disposizione funzionalità di base che favoriscono la mobilità degli utenti. SIP, inoltre, garantisce un elevato livello di flessibilità in quanto lavora in modo totalmente incorrelato al tipo di risorse applicative e di rete descritte nei documenti che incapsula.

Dal momento che la riservazione delle risorse di rete avviene parallelamente alla fase di segnalazione applicativa, questa architettura realizza di fatto la fornitura di

servizi di rete “on-demand” con precise caratteristiche di connettività e qualità del servizio.

Occorre considerare che la validità dell’architettura qui proposta è del tutto generale e non dipende dalla particolare scelta dell’infrastruttura di rete di trasporto.

2.5 Limiti

Un limite posto dall’architettura descritta in questa tesi è quello di non supportare una vera e propria negoziazione delle risorse di rete. Infatti la sessione viene stabilita dopo lo scambio di una richiesta e una risposta SIP, mentre una negoziazione necessiterebbe di un maggior numero di messaggi scambiati da parte degli User Agent coinvolti. Una segnalazione iniziale più ricca è realizzabile, ma solo attraverso l’introduzione di opportune estensioni alla segnalazione standard del protocollo SIP.

Capitolo 3

Elementi costitutivi dell'architettura

In questo capitolo vengono descritti gli elementi che stanno alla base dell'architettura presentata nel capitolo precedente. In primo luogo viene descritta l'architettura di rete NGN così come raccomandato dalle specifiche ITU-T, quindi sono descritte le due tecnologie su cui è incentrato il lavoro di tesi, ovvero il protocollo SIP e il framework RDF.

3.1 Next-Generation Network (NGN)

Data l'odierna natura della rete Internet, è lecito aspettarsi che le reti del futuro saranno basate su sistemi IP. In virtù di questo, i massimi organismi in materia di telecomunicazioni sono concordi nell'affermare che una rete di nuova generazione (next-generation network – NGN), costituisca una rete IP di tipo avanzato. Inoltre è generalmente inteso che la principale differenza fra i servizi di telecomunicazione tradizionali e una NGN è rappresentata dal passaggio da una serie di reti dedicate per applicazioni specifiche a una rete unica in grado di offrire ogni tipo di servizio (da quello telefonico tradizionale ad ogni tipo di servizio multimediale)[2].

La definizione formale di NGN contenuta nella raccomandazione ITU-T Y.2001[6] è la seguente:

> Next Generation Network (NGN): *a packet-based network able to provide telecommunication services and able to make use of multiple broadband QoS-enabled transport technologies and in which service-related*

> *functions are independent from underlying transport-related technologies. It enables unfettered access by users to networks and competing service providers and/or services of their choice. It supports generalized mobility which will allow consistent and ubiquitous provision of services to users*.

Nella definizione citata, è chiaramente enfatizzata la separazione fra i concetti di servizio e trasporto di rete, inoltre viene rimarcata la fondamentale importanza della QoS nel trasporto IP. Nella seconda frase si fa riferimento alla capacità di offrire servizi ad alto contenuto innovativo e alla possibilità per i service provider di pubblicare i propri servizi verso la rete, e per gli utenti di scegliere liberamente di quali servizi usufruire. Questo comporta un elemento di rottura con il passato perché sancisce la separazione fra gestore della rete e fornitore di servizi, in una visione di mercato libero. L'ultima frase comprende la parte di estensione per la mobilità degli utenti.

3.1.1 Architettura generale

L'architettura di una NGN si compone di due macroblocchi: lo strato di servizio (Service Stratum) e quello di trasporto (Transport Stratum). Sono poi previste opportune soluzioni che consentono l'interfacciamento con la rete da parte degli utenti (User-to-Network Interface), da parte dei fornitori di servizi applicativi (Application-to-Network Interface) ed infine di altre reti, di qualunque genere (Network-to-Network Interface).

Nel seguito si darà una concisa descrizione dei principali compiti assolti da ciascuno dei blocchi funzionali rappresentati in fig. 2.

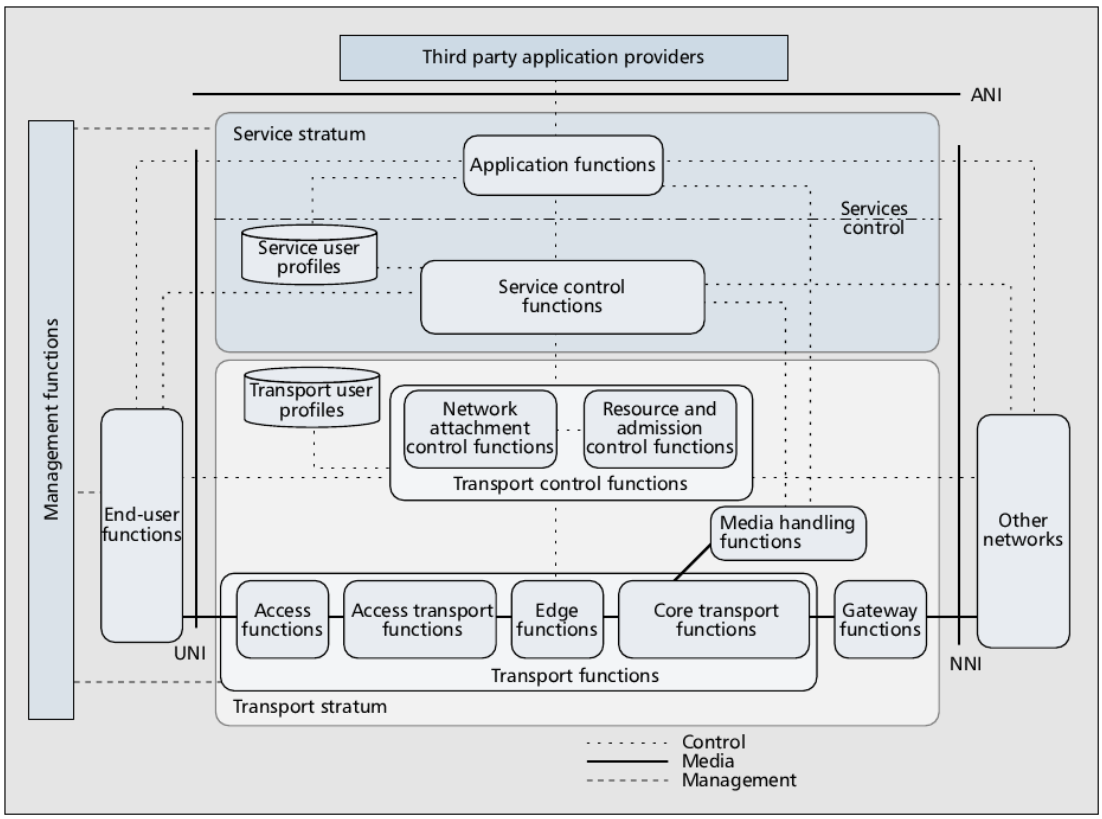

Figura 2. Architettura generale di una NGN

3.1.1.1 Transport Stratum

Le funzioni presenti all'interno dello strato di trasporto hanno il compito di fornire connettività IP sia agli apparati utente esterni alla rete, che ai server e agli altri dispositivi interni alla NGN.

Fra le responsabilità principali di questo strato, nel suo complesso, ricade quella di garantire la QoS end-to-end, ovvero da un capo all'altro delle connessioni.

3.1.1.2 Access Functions

Le funzioni di accesso permettono l'accesso alla rete da parte degli utenti. Si tratta di funzioni "technology-dependent", in quanto dipendono strettamente dalla tecnologia

di accesso utilizzata. Esse includono tecnologie di accesso radio (es. W-CDMA), e tecnologie di accesso cablato (xDSL, cavo coassiale, Ethernet, fibra ottica, ecc.)

3.1.1.3 Access Transport Functions

Queste funzioni si occupano del trasporto delle informazioni attraverso la rete di trasporto di accesso. Comprendono meccanismi di controllo della QoS che trattano direttamente il traffico utente: l'accodamento e lo scheduling dei pacchetti; la loro classificazione e marcatura; packet filtering; traffic shaping e traffic policing.

3.1.1.4 Edge Functions

Sono quelle funzioni che svolgono il processamento del traffico nel momento in cui il traffico di accesso viene immesso nella rete core.

3.1.1.5 Core Transport Functions

Sono le funzioni responsabili del trasporto dell'informazione su tutta la parte core della rete NGN. Dispongono di funzionalità di diversificazione della qualità del trasporto, che si basano su una forte interazione con le funzioni di controllo del trasporto (Transport Control Functions). Anche queste funzioni comprendono meccanismi di controllo della QoS e di gestione del traffico utente: gestione delle discipline di accodamento, scheduling dei pacchetti, classificazione e marcatura del traffico, packet filtering, traffic shaping e traffic policing.

3.1.1.6 Network Attachment Control Functions

Le NACF fanno parte delle funzioni di controllo del trasporto. In particolare gestiscono la registrazione degli utenti al livello di accesso e si occupano dell'inizializzazione di quelle funzioni che permettono agli utenti la fruizione di servizi NGN. Operano l'identificazione e l'autorizzazione a livello di rete, gestiscono gli spazi di indirizzamento IP della rete di accesso ed eseguono l'autenticazione delle sessioni di accesso. Inoltre assistono le funzioni utente nella fase di registrazione e di inizio dell'utilizzo della NGN.

3.1.1.7 Resource and Admission Control Functions

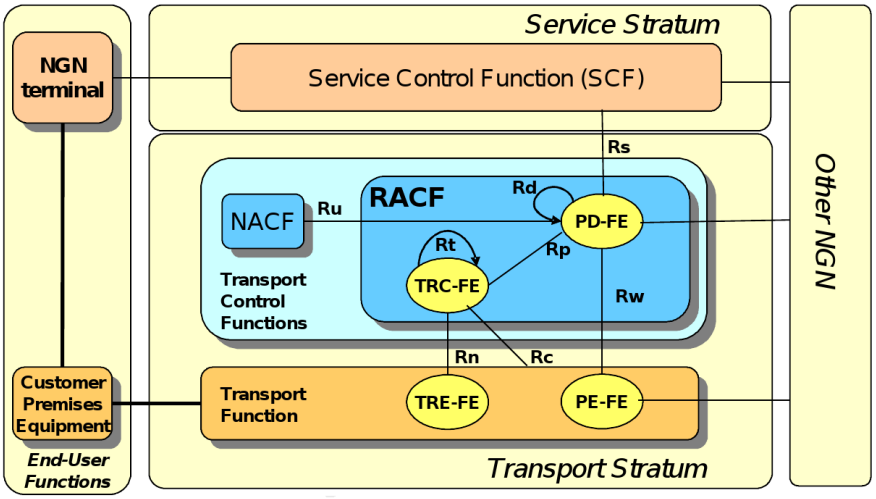

Figura 3. Dettaglio delle Resource and Admission Control Functions
PD-FE = Policy Decision Functional Entity
PE-FE = Policy Enforcement Functional Entity
TRC-FE = Transport Resource Control Functional Entity
TRE-FE = Transport Resource Enforcement Functional Entity

Le RACF sono le funzioni di controllo dello strato di trasporto che eseguono il controllo dell'autenticazione e dei privilegi degli utilizzatori della rete in base ai profili utente, alle politiche messe in atto dai service provider e alla disponibilità delle risorse di rete. Inoltre, queste funzioni sono preposte alla verifica di ammissibilità delle richieste di riservazione di risorse, controllando la disponibilità di risorse di rete.

Le RACF forniscono allo strato di servizio NGN una visione astratta di quella che è l'infrastruttura della rete di trasporto e permettono ad esso di controllare le funzionalità di rete come packet filtering, classificazione del traffico, marcatura e policing, riservazione e allocazione di banda, Network Address and Port Translation (NAPT), anti-spoofing degli indirizzi IP, NAPT/Firewall traversal e misurazione dell'utilizzo delle risorse di trasporto.

3.1.1.8 Transport User Profiles

Possono essere visti come un insieme di database cooperanti con funzionalità distribuite su tutta la rete NGN. Vengono utilizzati per la memorizzazione e la consultazione dei profili utente relativi alle funzionalità di trasporto.

3.1.1.9 Gateway Functions

Queste funzioni rendono possibile l'interconnessione della NGN con altre reti di natura differente (es. PSTN, ISDN, Internet) e con NGN appartenenti a differenti domini amministrativi.

3.1.1.10 Service Stratum

Questo strato offre servizi basati sulla sessione e non. Questi comprendono servizi di sottoscrizione e notifica di informazioni sulla presenza e meccanismi di scambio di messaggi per servizi di messaggistica istantanea.

Lo strato di servizio offre anche tutte le funzionalità associate ai servizi tradizionali, quali PSTN/ISDN, e le interfacce (hardware e software) verso gli utenti.

3.1.1.11 Service Control Functions

Fanno parte di queste funzioni quelle di controllo di sessione, quelle di registrazione, autenticazione e autorizzazione al livello di servizio. Queste funzioni possono includere funzioni di controllo delle risorse multimediali.

3.1.1.12 Service User Profiles

Sono il parallelo dei profili utente dello strato di trasporto. Rappresentano un insieme di database cooperanti con funzioni presenti su tutta la rete NGN, che mantengono le informazioni di profilo degli utenti a livello di servizio.

3.1.1.13 Application Functions

Sono le funzionalità messe a disposizione dalla rete stessa e dai service provider per gli utenti della rete NGN.

3.1.1.14 End-User Functions

Le interfacce verso gli utenti sono sia di tipo fisico che funzionale. Non viene specificato nulla riguardo le possibili interfacce o reti utilizzate dagli utenti di una NGN. La rete NGN fornisce supporto per qualsiasi categoria di dispositivo, da un semplice terminale telefonico ad una linea, a reti aziendali di qualsiasi complessità. Inoltre, i dispositivi degli utenti possono essere sia fissi che mobili.

3.2 Session Initiation Protocol (SIP)

SIP è un protocollo di segnalazione che opera al livello di sessione nel modello di riferimento ISO/OSI[7], collocandosi al livello applicativo nel modello TCP/IP[8]. La sua standardizzazione avvenne attraverso l'RFC 2543[9] (1999), successivamente aggiornato dall'RFC 3261[3] (2002).

Il SIP non è un protocollo che fornisce servizi, ma piuttosto mette a disposizione una serie di primitive sulle quali possono essere creati servizi. Le più importanti primitive offerte dal SIP, e dalle sue numerose estensioni più o meno standard, riguardano la localizzazione degli endpoint, definiti User Agent (UA), la loro registrazione in domini amministrativi, la possibilità di creare, modificare e terminare sessioni di comunicazione condivise e multiutente, la possibilità di registrarsi ed essere notificati sull'occorrenza di eventi asincroni.

Questo protocollo è stato pensato per incapsulare, nel corpo dei propri messaggi, la descrizione delle sessioni che gli User Agent intendono stabilire. Così facendo, le applicazioni SIP-enabled possono accordarsi, ad esempio, sull'uso di particolari protocolli di trasporto, codec multimediali e indirizzi ai quali spedirsi i pacchetti. Il protocollo maggiormente usato nella descrizione di sessioni multimediali è il Session Description Protocol (SDP).

Nonostante abbia trovato ampio uso all'interno di applicazioni prettamente multimediali, il protocollo SIP lavora in maniera del tutto indipendente rispetto a quella che è la tipologia di sessione che gestisce e dal protocollo di trasporto usato (UDP, TCP, TLS, SCTP).

Gli Uniform Resource Identifier (URI) SIP hanno la caratteristica di contenere una parte di identificativo dell'utente (o risorsa), che è fissa, e una parte di indirizzo, che è

variabile. Questo consente di fornire un intrinseco supporto alla mobilità degli utenti.

Il SIP fornisce un esteso supporto alla sicurezza dei dati, prevedendo meccanismi che includono la prevenzione di attacchi di tipo Denial-of-Service (DoS), l'autenticazione utente-utente e proxy-utente, la cifratura e il mantenimento della confidenzialità dei dati.

Il funzionamento del protocollo SIP è basato su tre punti cardine: transazione, dialogo e sessione.

3.2.1 Transazione

Il SIP è un protocollo basato sullo scambio di messaggi, secondo il modello richiesta-risposta. Lo scambio di messaggi a partire dalla prima richiesta fino alla risposta definitiva a tale richiesta è detto "transazione SIP"[3]. A dispetto del tradizionale modello richiesta-risposta, le transazioni SIP molto spesso non sono composte esclusivamente da due messaggi. Sono definite, infatti, diverse classi di risposta, ma queste possono essere raggruppate in risposte definitive e risposte provvisorie. Una transazione può comprendere una richiesta e un numero qualsiasi di risposte: una delle quali (l'ultima in ordine cronologico) definitiva, che sancisce la fine della transazione.

Esiste un caso eccezionale: nella transazione `INVITE` (chiamata in questo modo perché originata da una richiesta di tipo `INVITE`) la risposta definitiva dello User Agent remoto deve essere sempre seguita da un messaggio `ACK` per la conferma (analogamente a quanto avviene nel TCP con il 3-way handshake[11], ma per motivi differenti). Il messaggio `ACK` è un vero e proprio messaggio di richiesta dopo il quale non ci si aspetta una risposta.

3.2.2 Dialogo

Un dialogo SIP è definito come un rapporto paritetico (peer-to-peer) che intercorre fra due User Agent e dura per un certo intervallo di tempo[3]. La creazione di un dialogo è determinata dalla risposta non negativa (anche provvisoria) ad una richiesta `INVITE`. In pratica, un dialogo SIP nasce una volta che sia stabilita la raggiungibilità delle due parti in gioco. A dialogo stabilito, ognuna delle due parti è libera di dare vita a nuove transazioni all'interno di quel dialogo.

Un dialogo già stabilito viene terminato con una transazione `BYE`.

3.2.3 Sessione

Il SIP utilizza la definizione di sessione data dall'SDP, infatti viene considerata come l'insieme di tutti i partecipanti ad una conversazione, compresi i canali di comunicazione coinvolti[3]. Requisito fondamentale per la creazione di una sessione è l'esistenza di un dialogo. Inoltre, alla terminazione di un dialogo corrisponde sempre la chiusura della sessione associata. Tuttavia, il SIP non specifica quando una generica sessione debba essere considerata aperta, né indica in che modo deve essere chiusa. Questo infatti dipende dal tipo di sessione che le applicazioni intendono instaurare.

3.2.4 Principali metodi di richiesta

I principali metodi usati nei messaggi SIP di richiesta sono i seguenti:

- `REGISTER`: per registrare presso un registrar server le informazioni necessarie ad essere raggiunti;
- `INVITE`, `ACK` e `CANCEL`: per la creare e annullare la creazione di dialoghi;
- `BYE`: per terminare i dialoghi;
- `OPTIONS`: per interrogare i server e scoprire le funzionalità supportate;
- `PRACK`: per confermare le risposte di tipo provvisorio[12];
- `PUBLISH`: per la pubblicazione di eventi[13];
- `SUBSCRIBE` e `NOTIFY`: per la notifica di eventi asincroni[14];
- `MESSAGE`: per servizi di messaggistica istantanea[15].

3.2.5 Principali elementi di una rete SIP

- **SIP registrar server**: è un particolare server che accetta richieste di tipo `REGISTER` e memorizza le informazioni contenute in tali richieste in un database facente capo al dominio di sua pertinenza. Permette la localizzazione degli utenti.
- **SIP location server**: è il server al quale uno user agent si rivolge per conoscere l'indirizzo di un utente con il quale intende comunicare. Esso fornisce in risposta le informazioni mantenute in un database aggiornato da un registrar server.
- **SIP proxy server**: è un elemento di rete che fa da intermediario e si comporta sia da server che da client, in maniera tale da eseguire richieste per conto di altri client. Il suo ruolo principale è l'instradamento, ovvero deve assicurarsi di trasmettere la richiesta ad un'entità più vicina (generalmente non in termini geografici) all'utente di destinazione. I proxy interpretano le informazioni contenute nell'intestazione dei messaggi, ne possono aggiungere di nuove e modificarne alcune, prima di inoltrarli.

3.3 Resource Description Framework (RDF)

RDF significa letteralmente "struttura per la descrizione di risorse". Fa parte delle raccomandazioni del consorzio W3C dal 1999[16]. In origine era pensato per essere utilizzato nella definizione dei metadati[1] nel Web, ma all'inizio degli anni 2000, fino al 2004, furono pubblicate nuove raccomandazioni[17] che lo fecero evolvere in un framework generale per la modellizzazione dell'informazione, nonché nello strumento designato dal W3C per lo scambio della conoscenza nel Web[18].

L'RDF è stato progettato per poter rappresentare qualunque tipo di informazione, nella maniera più flessibile e libera possibile. La sua forza risiede nella sua generalità.

La grande espressività dell'RDF deriva da una semantica di base molto formale, che consente di definire in modo rigoroso il significato di ogni espressione.

L'RDF è strettamente imparentato con l'Extensible Markup Language (XML)[19]. Infatti utilizza tutti i tipi di dato XML, definisce uno speciale tipo di dato che con-

[1] Metadati: letteralmente, dati che riguardano altri dati.

sente l'incapsulamento di documenti XML direttamente in documenti RDF, e infine il metodo di serializzazione predefinito per i grafi RDF è basato sulla sintassi XML. Un documento RDF rappresentato secondo la sintassi XML è detto "documento RDF/XML".

3.3.1 Triple

L'RDF dà la possibilità di fare affermazioni su qualsiasi cosa. Ogni affermazione (o RDF statement) è sempre composta da tre elementi: il soggetto, il predicato (o proprietà) e l'oggetto. Per questo motivo uno statement viene anche chiamato "tripla RDF".

Una tripla rappresenta l'affermazione di una relazione che intercorre tra il soggetto e l'oggetto, e può essere rappresentata in maniera grafica per mezzo di due nodi uniti da un arco orientato. L'orientamento dell'arco – che rappresenta il predicato – è significativa e va sempre nella direzione dell'oggetto.

Un insieme di triple è detto "grafo RDF", pertanto un grafo RDF corrisponde all'affermazione di un insieme di relazioni fra risorse.

3.3.2 Vocabolari

L'RDF dispone delle primitive necessarie e sufficienti per poter esprimere semplici affermazioni riguardo qualsiasi risorsa. Tuttavia non prevede, di per sé, alcun meccanismo per la definizione degli oggetti da descrivere e delle proprietà usate per descriverli. Per fare ciò è necessario fare ricorso all'RDF Vocabulary Description Language[20], o RDF Schema.

L'RDF Schema mette a disposizione i meccanismi necessari per la formulazione di vocabolari RDF. In particolare, consente di descrivere gruppi di risorse e le relazioni che le legano.

Il sistema utilizzato dall'RDF Schema si poggia sui concetti di "classe" e "proprietà" in maniera simile ai linguaggi di programmazione object-oriented, come Java™; tuttavia presenta una sostanziale differenza, in quanto non descrive una classe di risorse in base alle proprietà che possiede, bensì descrive una proprietà in base alle classi di risorse alle quali si applica. È questo il ruolo dei concetti di "domain" e "range": con

domain si specifica la classe di risorse che possiede la proprietà in questione, mentre con range si indica la classe di risorse a cui fa riferimento la proprietà in questione. Ad esempio, se si dovesse descrivere un libro, si definirebbero le classi `Libro` e `Persona`, e quindi la proprietà `autore`; tale proprietà sarebbe caratterizzata dall'avere per dominio la classe `Libro` e per range (il codominio di una funzione, in termini matematici) la classe `Persona`.

Questo metodo ha la caratteristica di essere "aperto", nella misura in cui permette a chiunque altro di definire altre proprietà che abbiano `Libro` come dominio e/o `Persona` come range, in modo immediato e semplice, senza dover ridefinire la descrizione originale di queste classi. Pertanto, l'approccio RDF rende possibile l'estensione della descrizione di risorse già esistenti, che è uno dei principi architeturali del Web[21].

Capitolo 4

Service Oriented Optical Network (SOON)

Service Oriented Optical Network (SOON)[22] è una infrastruttura che fa da tramite nell'interazione tra applicazioni e rete GMPLS, per la fornitura di servizi di rete a larghissima banda e nel rispetto di precisi parametri di QoS. L'architettura SOON implementa un piano di servizio, o Service Plane (SP), concettualmente posto al di sopra del Control Plane (CP) GMPLS. Grazie al SP, SOON esegue una separazione netta fra quelle che sono le funzionalità di trasporto di rete e quelli che sono i servizi che la rete offre. In particolare, questa soluzione permette alla rete di trasporto di essere indipendente rispetto all'introduzione di nuovi servizi o all'aggiornamento di quelli esistenti, e permette al CP di non doversi curare degli aspetti legati alla composizione dei servizi di rete, ma di concentrarsi sulla fornitura di connettività.

4.1 Service Plane (SP)

Il SP sfrutta la segnalazione GMPLS per fornire servizi di trasporto con un livello di astrazione tale da poter essere invocati direttamente dalle applicazioni utenti della rete, infatti essi permettono di specificare i parametri in termini di spazi di indirizzamento degli end-host e di qualità del servizio percepita end-to-end.

Il SP è suddiviso in due livelli funzionali:

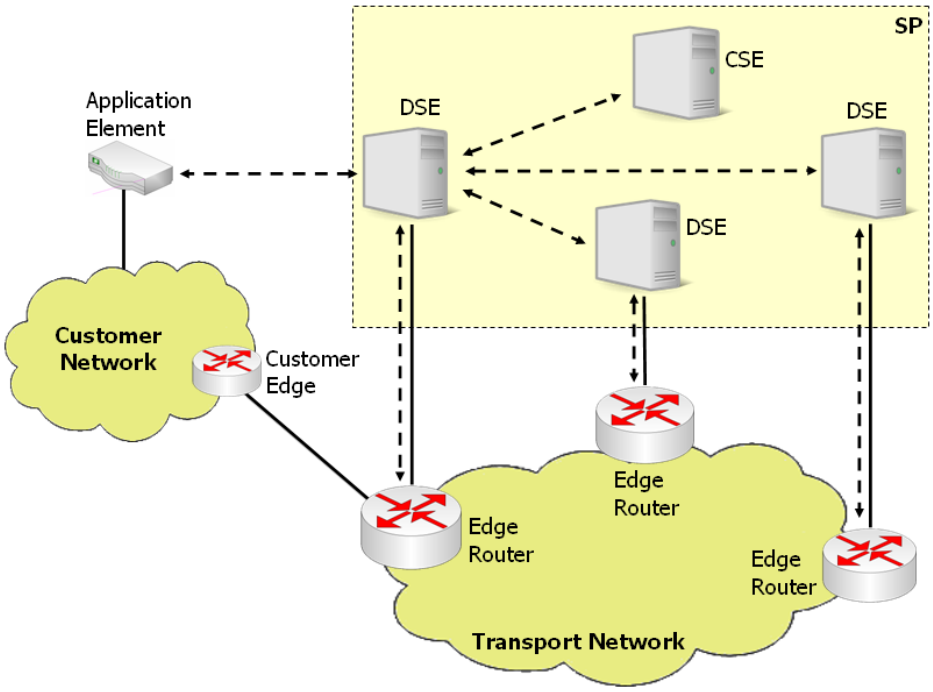

Figura 4. Architettura SOON

- **Centralized Service Layer (CSL)**: mantiene il Service Level Agreement (SLA) è responsabile degli aspetti commerciali, della gestione dei profili di utente e all'autorizzazione delle richieste di servizio. Questo livello è implementato da un Centralized Service Element (CSE).

- **Distributed Service Layer (DSL)**: si occupa degli aspetti dinamici della fornitura di servizio, come il controllo della disponibilità di particolari servizi/risorse e l'effettiva fornitura del servizio. Questo livello è implementato da una serie di Distributed Service Element (DSE) che, grazie ad una segnalazione distribuita, riescono a soddisfare le richieste delle applicazioni. I servizi sono effettivamente attivati accedendo direttamente al Control Plane GMPLS, per questo motivo il DSL ha una struttura che ricalca quella del CP. In pratica vi è un DSE per ogni router di bordo della rete di trasporto.

4.1.1 Distributed Service Element (DSE)

Il DSE riceve e processa le richieste di servizio fatte dalle applicazioni. Per soddisfare tali richieste, esso scatena uno scambio di messaggi di segnalazione con il CSE per autenticare e autorizzare le applicazioni, e con gli altri DSE coinvolti nel servizio per accordarsi sulle impostazioni con le quali verranno configurati i router di bordo. Dopodiché ogni DSE utilizzerà le primitive offerte dal router che controlla. Queste primitive comprendono i comandi di configurazione, come la creazione di un circuito, e il monitoraggio dello stato delle risorse, come il controllo della disponibilità di banda. Il DSE è in grado di stabilire quali altri DSE sono coinvolti nella fornitura di un particolare servizio elaborando le informazioni contenute nel Network Topology Resource Database (NRTD). Questo database è di tipo distribuito, in quanto presente in ogni DSE, e viene periodicamente aggiornato attraverso la cosiddetta segnalazione di background. Esso contiene gli indirizzi di tutti i DSE e le informazioni sulle interfacce dei router che essi controllano.

4.2 SOON e NGN

Il modello architetturale SOON si presta molto favorevolmente ad essere utilizzato come base per reti conformi alle specifiche ITU-T NGN[2] grazie alla scomposizione fra servizio e trasporto che opera. Il Distributed Service Layer del Service Plane SOON, ovvero l'insieme dei DSE che lo compongono, può essere visto, dal punto di vista funzionale, come l'insieme delle funzioni RACF e NACF in una NGN. Infatti il DSL opera il controllo della rete di trasporto sottostante, nascondendo alle applicazioni i dettagli tecnologici e topologici, ed esegue l'autorizzazione delle applicazioni a livello di rete. Questo suggerisce la possibilità di introdurre un Service Stratum logicamente posto al di sopra del SP SOON, comunicante con quest'ultimo attraverso un'opportuna interfaccia (R_S), come illustrato in figura 5.

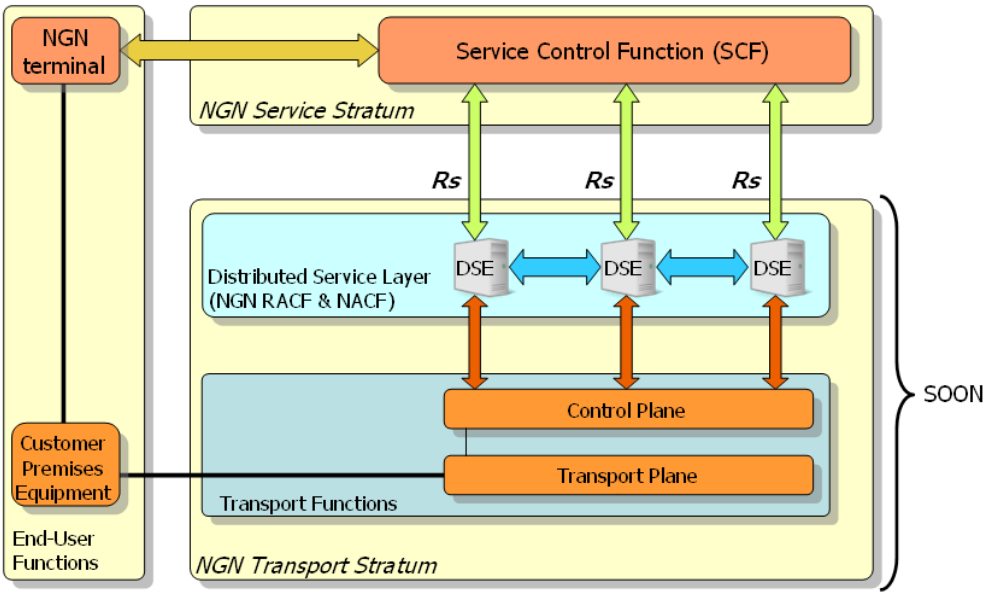

Figura 5. Rete NGN supportata dall'architettura SOON

Capitolo 5

Lavoro svolto

Il lavoro di questa tesi rappresenta un primo passo verso la realizzazione dell'architettura di application-aware networking su rete NGN proposta nel capitolo 2. Per l'implementazione del Transport Stratum NGN si è deciso di avvalersi dell'opportunità offerta dall'infrastruttura di rete SOON, in quanto si presta favorevolmente alla costituzione di reti conformi con le linee guida ITU-T NGN, come già sottolineato nel capitolo precedente. Per quanto concerne il Service Stratum, in particolare il modulo SIP-M, è stato implementato attraverso un proxy server SIP. In particolare, si è scelto di utilizzare il software OpenSIPS[1] (versione 1.4.2) per l'elevata maturità raggiunta nello sviluppo e per la sua architettura modulare e aperta (open source), che si presta ad un uso altamente personalizzato e supporta l'introduzione di funzionalità aggiuntive.

Il lavoro svolto si suddivide in tre parti:

1. definizione del vocabolario RDF per la descrizione delle risorse di rete: Network Resource Description Language (NRDL);
2. sviluppo dell'interfaccia fra SIP-M e NET-M, ovvero di un modulo software per OpenSIPS;
3. sviluppo del modulo NET-M, ovvero un software con il compito di tradurre le richieste ad alto livello di riservazione delle risorse di rete (espresse secondo il vocabolario NRDL), in richieste di servizio atomiche interpretabili dagli apparati DSE della rete SOON.

[1] OpenSIPS – Open SIP Server, `http://www.opensips.org/`

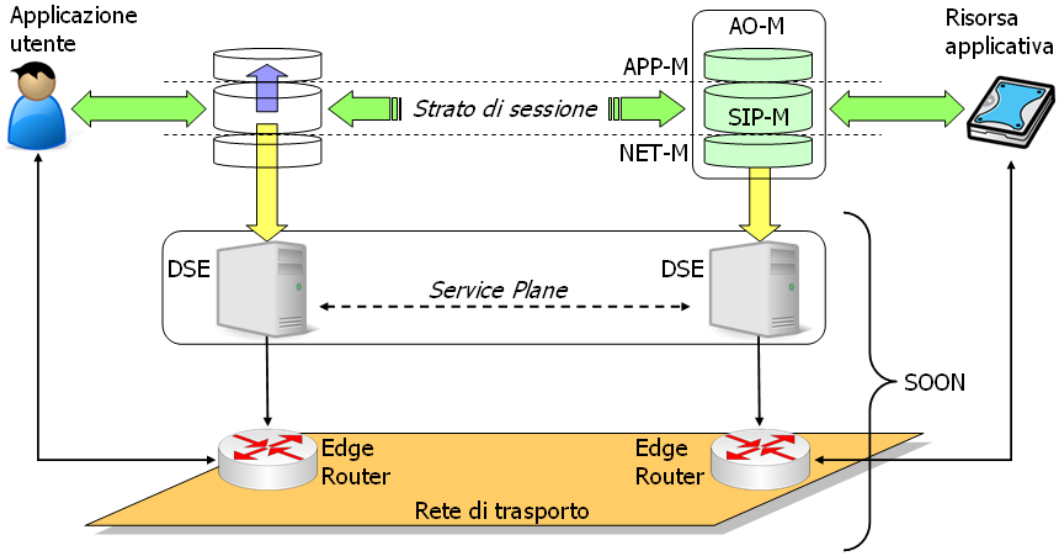

Figura 6. Infrastruttura SOON usata come Transport Stratum per l'architettura proposta

Questo lavoro è stato portato avanti con il fine ultimo di realizzare una semplice struttura di prova, un testbed (descritto nel prossimo capitolo), attraverso cui poter validare sperimentalmente l'architettura di application-aware networking proposta. A questo scopo è stato inoltre realizzato un software che implementa i SIP User Agent coinvolti nella segnalazione.

5.1 Network Resource Description Language (NRDL)

NRDL è stato creato con l'obiettivo di fornire un supporto alla descrizione di risorse di rete, in particolar modo per lo scambio di richieste e risposte riguardanti la loro riservazione. Esso stabilisce:

- le classi di risorse che è possibile descrivere;
- le particolari risorse di rete di interesse;
- le proprietà di ciascuna classe: che possono essere parametri o relazioni con altre classi.

NRDL si avvale di una descrizione di tipo aperto, in accordo con la visione RDF; di conseguenza è al di fuori dei suoi scopi stabilire i seguenti fatti:

- le regole di composizione, ovvero la quantità e la qualità delle proprietà che una classe deve o può possedere;
- i valori (e le loro eventuali unità di misura) che una particolare proprietà/parametro deve o può assumere.

Ad esempio, questo vocabolario non impone l'obbligo per la classe `Endpoint` di avere esattamente un parametro `ipv4` ed un parametro `port`. Dovranno essere le applicazioni che usano il vocabolario a stabilire questo tipo di regole e accertarsi del loro rispetto. Il medesimo discorso vale per i possibili valori degli attributi delle classi.

Il vocabolario è definito nella maniera standard specificata dall'RDF Schema, infatti è descritto come "ontologia" OWL[2] ed è contenuto all'interno di un regolare documento RDF/XML. Ad ogni vocabolario deve essere associato un URI di riferimento, che sarà il namespace XML utilizzato nei documenti che useranno tale vocabolario; quello scelto per NRDL – del tutto fittizio – è il seguente:

`http://deisnet.deis.unibo.it/nrdl/nrdl-syntax-ns#`

Di seguito sono riportati l'inizio del documento e la dichiarazione di ontologia:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax
   -ns#"
 xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
 xmlns:owl="http://www.w3.org/2002/07/owl#"
 xmlns:dc="http://purl.org/dc/elements/1.1/">

<rdf:Description rdf:about="http://deisnet.deis.unibo.it/
   nrdl/nrdl-syntax-ns#">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#
     Ontology"/>
  <rdfs:label>nrdl</rdfs:label>
  <dc:title xml:lang="en">Network Resource Description
     Language (NRDL)</dc:title>
```

[2]OWL: Web Ontology Language, `http://www.w3.org/TR/owl-features/`

```
  <dc:description xml:lang="en">A vocabulary suitable to
     describe network resources, their relationships and
     properties.</dc:description>
  <owl:versionInfo>0.1</owl:versionInfo>
</rdf:Description>
```

5.1.1 Risorsa fisica

È definita dall'identificativo `PhysicalResource`. Per risorsa fisica si intende qualunque apparato di rete in grado di comunicare con altri. Questo tipo di risorsa è utilizzato per scopi di identificazione, ad esempio per indicare la sorgente di una richiesta e i nodi intermedi che questa ha attraversato.

Definizione di risorsa fisica:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#PhysicalResource">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">Physical resource</rdfs:label>
  <rdfs:comment xml:lang="en">A physical network resource
     .</rdfs:comment>
</rdfs:Class>
```

Sono state definite tre sottoclassi:

1. `DSE`;
2. `SIPDevice`;
3. `SIPProxy`.

5.1.1.1 Classe `DSE`

Questa classe può essere usata per descrivere gli omonimi apparati dell'infrastruttura SOON. È definita per mezzo della seguente dichiarazione:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#DSE">
```

```
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">DSE (Distributed Service
     Element)</rdfs:label>
  <rdfs:comment xml:lang="en">A DSE (Distributed Service
     Element) of the SOON (Service-Oriented Optical Network
     ) architecture.</rdfs:comment>
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#PhysicalResource"/>
</rdfs:Class>
```

5.1.1.2 Classe `SIPDevice`

Questa classe è utilizzata per la descrizione di qualsiasi dispositivo SIP che sia utilizzatore della rete. È così definita:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#SIPDevice">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">SIP device</rdfs:label>
  <rdfs:comment xml:lang="en">A NRDL-enabled SIP device.</
     rdfs:comment>
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#PhysicalResource"/>
</rdfs:Class>
```

5.1.1.3 Classe `SIPProxy`

Questa classe è usata per descrivere i nodi SIP intermedi attraversati dai messaggi di segnalazione. É stata definita con la seguente dichiarazione:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#SIPProxy">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">SIP proxy server</rdfs:label>
  <rdfs:comment xml:lang="en">An RDF enabled SIP proxy.</
     rdfs:comment>
```

```
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#PhysicalResource"/>
</rdfs:Class>
```

5.1.2 Risorsa logica

Questa classe è identificata per mezzo dell'identificativo `LogicalResource`. Per risorsa logica è intesa qualsiasi risorsa riconducibile alli scambio di dati fra due o più utenti della rete.

Definizione di risorsa logica:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#LogicalResource">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">Logical resource</rdfs:label>
  <rdfs:comment xml:lang="en">A logical network resource
     such as dialog, session and connection.</rdfs:comment>
</rdfs:Class>
```

Sono definite le seguenti risorse logiche (di primo livello):

- `Endpoint`;
- `Connection`;
- `Session`;
- `Dialog`;
- `TransportParameter`.

5.1.2.1 Classe `Endpoint`

Questa classe definisce il capo di una connessione. È univocamente identificato nella rete dall'accoppiata formata da indirizzo IP e porta. La classe è definita tramite la seguente dichiarazione:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#Endpoint">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">Endpoint</rdfs:label>
  <rdfs:comment xml:lang="en">Network endpoint (address and
      port number).</rdfs:comment>
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#LogicalResource"/>
</rdfs:Class>
```

5.1.2.2 Classe `Connection`

Questa classe è utilizzata per descrivere una connessione. In NRDL per "connessione" si intende lo scambio di dati fra due elementi di rete. In particolare, nella definizione di questo alfabeto si è scelto di vedere una connessione come un segmento orientato che va da un endpoint "sorgente" ad uno "destinazione". Questa scelta consente un alto grado di flessibilità in quanto, oltre a consentire la descrizione di trasferimenti di dati monodirezionali, permette la descrizione di trasferimenti di dati bidirezionali di tipo asimmetrico, ovvero nei quali i dati che fluiscono in un verso hanno proprietà differenti da quelli che fluiscono nel verso opposto.

Questa classe è definita tramite la seguente dichiarazione:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#Connection">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">Connection</rdfs:label>
  <rdfs:comment xml:lang="en">A logical connection between
     two network elements.</rdfs:comment>
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#LogicalResource"/>
</rdfs:Class>
```

5.1.2.3 Classe `Session`

Questa classe descrive una sessione di comunicazione. Per "sessione di comunicazione" si intende un insieme di connessioni che sussistono fra due elementi di rete.

Definizione della classe:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#Session">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">Session</rdfs:label>
  <rdfs:comment xml:lang="en">A logical connection between
     two network elements.</rdfs:comment>
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#LogicalResource"/>
</rdfs:Class>
```

5.1.2.4 Classe `Dialog`

Questa classe descrive un dialogo fra entità di rete. Per "dialogo" si intende un insieme di sessioni fra un numero qualsiasi di elementi di rete.

Definizione della classe:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#Dialog">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">Dialog</rdfs:label>
  <rdfs:comment xml:lang="en">A logical dialog among
     several network elements.</rdfs:comment>
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#LogicalResource"/>
</rdfs:Class>
```

5.1.2.5 Classe `TransportParameter`

È una classe "contenitore" proprio come `PhysicalResource` e `LogicalResource`. Essa rappresenta un'astrazione di un generico parametro di qualità del trasporto.

Definizione:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#TransportParameter">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">Transport parameter</rdfs:label
     >
  <rdfs:comment xml:lang="en">A parameter pertaining to the
      transport of data between two network elements.</rdfs
     :comment>
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#LogicalResource"/>
</rdfs:Class>
```

Sono definiti i seguenti parametri di trasporto:

- `Bandwidth`;
- `TrafficClass`;
- `MaxDelay`;
- `MaxJitter`;
- `MaxPacketLoss`.

5.1.2.6 Classe `Bandwidth`

Questa classe rappresenta la larghezza di banda (intesa come bitrate) di una connessione. Definizione:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#Bandwidth">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">Bandwidth</rdfs:label>
  <rdfs:comment xml:lang="en">Value of bandwidth related to
      a connection between two network elements.</rdfs:
     comment>
```

```
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#TransportParameter"/>
</rdfs:Class>
```

5.1.2.7 Classe `TrafficClass`

Questa classe rappresenta la classe di traffico nella quale rientrano i dati relativi ad una connessione. Definizione:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#TrafficClass">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">Traffic class</rdfs:label>
  <rdfs:comment xml:lang="en">Traffic class related to a
     connection between two network elements.</rdfs:comment
     >
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#TransportParameter"/>
</rdfs:Class>
```

5.1.2.8 Classe `MaxDelay`

Questa classe rappresenta il massimo ritardo di trasmissione da un capo all'altro di una connessione. Definizione:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#MaxDelay">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">Maximum delay</rdfs:label>
  <rdfs:comment xml:lang="en">The maximum value of the
     transfer delay.</rdfs:comment>
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#TransportParameter"/>
</rdfs:Class>
```

5.1.2.9 Classe `MaxJitter`

Questa classe rappresenta il massimo "jitter" di una connessione. Dove per jitter si intende lo scostamento fra i valori massimo e minimo del ritardo di trasmissione. Definizione:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#MaxJitter">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">Maximum jitter</rdfs:label>
  <rdfs:comment xml:lang="en">The maximum value of the
     jitter (variation between maximum and minimum delay)
     .</rdfs:comment>
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#TransportParameter"/>
</rdfs:Class>
```

5.1.2.10 Classe `MaxPacketLoss`

Questa classe rappresenta il valore massimo della probabilità di perdita di pacchetti in un a connessione. Definizione:

```
<rdfs:Class rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#MaxPacketLoss">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">Maximum packet loss</rdfs:label
     >
  <rdfs:comment xml:lang="en">The maximum packet loss
     probability.</rdfs:comment>
  <rdfs:subclassOf rdf:resource="http://deisnet.deis.unibo.
     it/nrdl/nrdl-syntax-ns#TransportParameter"/>
</rdfs:Class>
```

5.1.3 Proprietà

Una volta definite le classi, si è passati alla definizione delle loro proprietà (o predicati, nella terminologia RDF).

5.1.3.1 Proprietà `name`

Questa proprietà permette di assegnare un identificativo, comprensibile agli umani, alle risorse rappresentate in un documento RDF. Questa proprietà può essere attribuita a qualsiasi classe definita all'interno del vocabolario. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#name">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">name</rdfs:label>
  <rdfs:comment xml:lang="en">A string representing an
     element in human-readable way.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#LogicalResource"/>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#PhysicalResource"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
     schema#Literal"/>
</rdf:Property>
```

5.1.3.2 Proprietà `duration`

Questa proprietà permette di assegnare una durata a risorse di classe `Connection`, `Session` e `Dialog`. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#duration">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">duration</rdfs:label>
  <rdfs:comment xml:lang="en">An integer representing the
     session duration.</rdfs:comment>
```

```
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Connection"/>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Session"/>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Dialog"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
     XMLSchema#integer"/>
</rdf:Property>
```

5.1.3.3 Proprietà `hasSession`

Questa proprietà consente di assegnare una sessione ad un dialogo. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#hasSession">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">has session</rdfs:label>
  <rdfs:comment xml:lang="en">The hasSession property
     assigns a session to a dialog.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Dialog"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
     schema#Session"/>
</rdf:Property>
```

5.1.3.4 Proprietà `hasConnection`

Questa proprietà assegna una connessione ad una sessione. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#hasConnection">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">has connection</rdfs:label>
  <rdfs:comment xml:lang="en">The hasConnection property
     assigns a connection to a session.</rdfs:comment>
```

```
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Session"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
     schema#Connection"/>
</rdf:Property>
```

5.1.3.5 Proprietà `source`

Questa proprietà assegna l'endpoint sorgente ad una connessione. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#source">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">source endpoint</rdfs:label>
  <rdfs:comment xml:lang="en">The endpoint from which the
     data stream is originated.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Connection"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
     schema#Endpoint"/>
</rdf:Property>
```

5.1.3.6 Proprietà `destination`

Questa proprietà assegna l'endpoint di destinazione ad una connessione. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#destination">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">destination endpoint</rdfs:
     label>
  <rdfs:comment xml:lang="en">The endpoint towards which
     the data stream is headed.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Connection"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/01/rdf-
     schema#Endpoint"/>
```

```
</rdf:Property>
```

5.1.3.7 Proprietà `protocol`

Questa proprietà permette di indicare il protocollo di trasporto utilizzato in una connessione. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#protocol">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">protocol</rdfs:label>
  <rdfs:comment xml:lang="en">The transport protocol used
     in a connection.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Connection"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
     XMLSchema#string"/>
</rdf:Property>
```

5.1.3.8 Proprietà `hasBandwidth`

Questa proprietà permette di specificare la bitrate associata ad una connessione. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#hasBandwidth">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">bandwidth</rdfs:label>
  <rdfs:comment xml:lang="en">The bandwidth (assumed as
     bitrate) related to a data transport.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Connection"/>
  <rdfs:range rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Bandwidth"/>
</rdf:Property>
```

5.1.3.9 Proprietà `hasTrafficClass`

Questa proprietà assegna una classe di traffico ad una connessione. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#hasTrafficClass">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">has traffic class</rdfs:label>
  <rdfs:comment xml:lang="en">The traffic class of data
     involved in a data transport.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Connection"/>
  <rdfs:range rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#TrafficClass"/>
</rdf:Property>
```

5.1.3.10 Proprietà `hasMaxDelay`

Questa proprietà assegna ad una connessione il massimo ritardo di trasmissione. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#hasMaxDelay">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">has maximum delay</rdfs:label>
  <rdfs:comment xml:lang="en">The maximum value of delay in
      a data transport.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Connection"/>
  <rdfs:range rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#MaxDelay"/>
</rdf:Property>
```

5.1.3.11 Proprietà `hasMaxJitter`

Questa proprietà assegna ad una connessione il massimo valore di jitter. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#hasMaxJitter">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">has maximum jitter</rdfs:label>
  <rdfs:comment xml:lang="en">The maximum value of jitter
     in a data transport.</rdfs:comment>
  <rdfs:domain
   rdf:resource="http://deisnet.deis.unibo.it/nrdl/nrdl-
      syntax-ns#Connection"/>
  <rdfs:range
   rdf:resource="http://deisnet.deis.unibo.it/nrdl/nrdl-
      syntax-ns#MaxJitter"/>
</rdf:Property>
```

5.1.3.12 Proprietà `hasMaxPacketLoss`

Questa proprietà assegna ad una connessione la massima probabilità di perdita di pacchetti. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#hasMaxPacketLoss">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">has maximum packet loss
     probability</rdfs:label>
  <rdfs:comment xml:lang="en">The maximum packet loss
     probability in a data transport.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Connection"/>
  <rdfs:range rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#MaxPacketLoss"/>
</rdf:Property>
```

5.1.3.13 Proprietà `ipv4`

Questa proprietà assegna un indirizzo IPv4 as un endpoint. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#ipv4">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">IPv4 address</rdfs:label>
  <rdfs:comment xml:lang="en">IPv4 address of a network
     endpoint.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Endpoint"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
     XMLSchema#string"/>
</rdf:Property>
```

5.1.3.14 Proprietà `port`

Questa proprietà assegna un numero di porta ad un endpoint. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#port">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">port</rdfs:label>
  <rdfs:comment xml:lang="en">The port number belonging to
     a network endpoint involved in a connection.</rdfs:
     comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Endpoint"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
     XMLSchema#integer"/>
</rdf:Property>
```

5.1.3.15 Proprietà `requirementStrength`

Questa proprietà assegna ad un parametro di trasporto (sottoclasse di `Transport-Parameter`) il livello di importanza che il richiedente associa alla riservazione della particolare risorsa di rete. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#requirementStrength">
```

```
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">requirement strength</rdfs:
     label>
  <rdfs:comment xml:lang="en">Requirement strength for the
     reservation of a specific transport parameter.</rdfs:
     comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#TransportParameter"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
     XMLSchema#string"/>
</rdf:Property>
```

5.1.3.16 Proprietà `answer`

Questa proprietà assegna ad un parametro di trasporto la risposta derivante dal suo tentativo di riservazione. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#answer">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">answer</rdfs:label>
  <rdfs:comment xml:lang="en">Answer representing the
     status of the reservation of a particular transport
     parameter.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#TransportParameter"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
     XMLSchema#string"/>
</rdf:Property>
```

5.1.3.17 Proprietà `bandwidthValue`

Questa proprietà assegna ad una risorsa di classe `Bandwidth` il valore di banda associato. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#bandwidthValue">
```

```
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">bandwidth value</rdfs:label>
  <rdfs:comment xml:lang="en">The bandwidth value related
     to a connection.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#Bandwidth"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
     XMLSchema#integer"/>
</rdf:Property>
```

5.1.3.18 Proprietà `trafficClassValue`

Questa proprietà assegna ad una risorsa di classe `TrafficClass` il valore associato. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#trafficClassValue">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">traffic class value</rdfs:label
     >
  <rdfs:comment xml:lang="en">A string representing the
     traffic class of a connection.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#TrafficClass"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
     XMLSchema#string"/>
</rdf:Property>
```

5.1.3.19 Proprietà `maxDelayValue`

Questa proprietà assegna ad una risorsa di classe `MaxDelay` il valore associato. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#maxDelayValue">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
```

```
  <rdfs:label xml:lang="en">maximum delay value</rdfs:label
     >
  <rdfs:comment xml:lang="en">An integer number
     representing the maximum transfer delay.</rdfs:comment
     >
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#MaxDelay"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
     XMLSchema#integer"/>
</rdf:Property>
```

5.1.3.20 Proprietà `maxJitterValue`

Questa proprietà assegna ad una risorsa di classe `MaxJitter` il valore associato. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#maxJitterValue">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
  <rdfs:label xml:lang="en">maximum jitter value</rdfs:
     label>
  <rdfs:comment xml:lang="en">An integer number
     representing the maximum jitter.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#MaxJitter"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
     XMLSchema#integer"/>
</rdf:Property>
```

5.1.3.21 Proprietà `maxPacketLossValue`

Questa proprietà assegna ad una risorsa di classe `MaxPacketLoss` il valore associato. Definizione:

```
<rdf:Property rdf:about="http://deisnet.deis.unibo.it/nrdl/
   nrdl-syntax-ns#maxPacketLossValue">
  <rdfs:isDefinedBy rdf:resource="http://deisnet.deis.unibo
     .it/nrdl/schema.rdf"/>
```

```
  <rdfs:label xml:lang="en">maximum packet loss probability
     </rdfs:label>
  <rdfs:comment xml:lang="en">A real number representing
     the maximum packet loss probability.</rdfs:comment>
  <rdfs:domain rdf:resource="http://deisnet.deis.unibo.it/
     nrdl/nrdl-syntax-ns#MaxPacketLoss"/>
  <rdfs:range rdf:resource="http://www.w3.org/2001/
     XMLSchema#float"/>
</rdf:Property>
```

5.2 Interfaccia SIP-M/NET-M

La seconda parte del lavoro è stata rivolta allo sviluppo di un modulo software per OpenSIPS che facesse da interfaccia fra i moduli SIP-M e NET-M dell'architettura proposta in questo lavoro di tesi (fig. 7). Il modulo per OpenSIPS è stato denominato "Netres", in quanto fa da mediatore nella riservazione delle risorse di rete.

Ogni modulo di OpenSIPS ha la possibilità di esportare parametri di configurazione e funzioni. L'esportazione dei parametri consiste nella possibilità di poterli impostare dall'interno del file di configurazione, mentre l'esportazione di funzioni consiste nella possibilità di usarle all'interno degli script di routing dei messaggi, sempre all'interno del file di configurazione.

Per mezzo del file di configurazione di OpenSIPS possono essere impostate opzioni come i protocolli di trasporto da utilizzare, le porte e gli indirizzi di ascolto, i moduli da caricare, i loro parametri e gli script di routing dei messaggi SIP.

Nella parte iniziale del file di configurazione devono essere specificati tutti i moduli che si intendono caricare. Ogni modulo mette a disposizione una particolare funzionalità, o un insieme di funzionalità strettamente collegate, perciò la scelta del caricamento dei moduli è funzione del particolare uso che si intende fare di OpenSIPS. Ad esempio, per caricare il modulo Netres occorre inserire la seguente riga:

```
loadmodule "netres.so"
```

Dopo aver deciso quali moduli verranno caricati, è possibile specificare i valori dei loro parametri e infine gli script di routing dei messaggi SIP. Questi script sono il luogo nel quale vengono prese le decisioni sull'instradamento dei messaggi (sia di

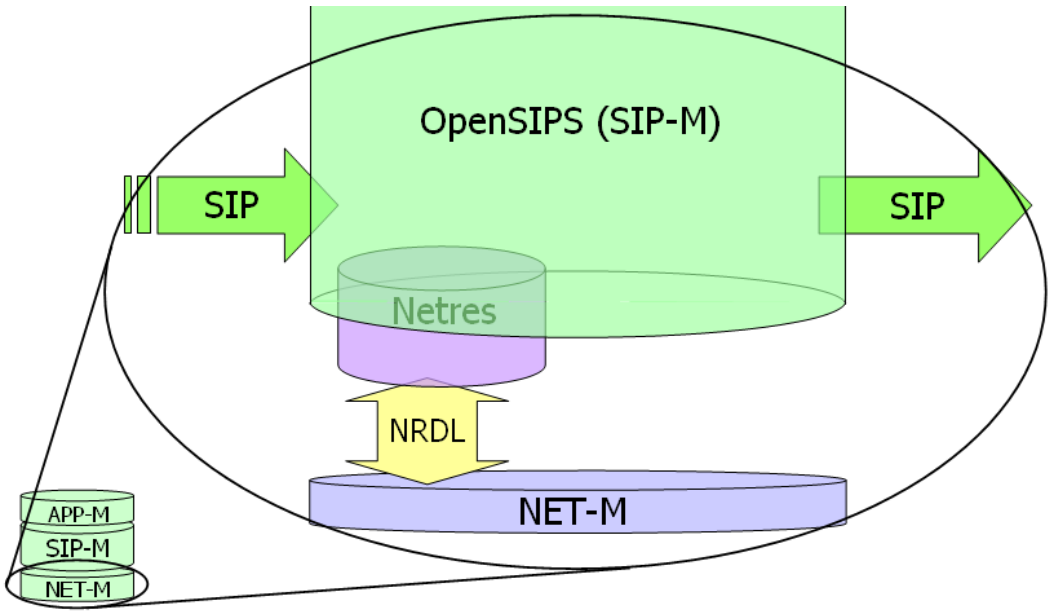

Figura 7. Interfaccia fra SIP-M e NET-M

richiesta che di risposta) e sul loro trattamento in genere. Qui vengono utilizzate le funzioni esportate dai moduli per ottenere il comportamento complessivo desiderato per l'applicazione.

5.2.1 Parametri esportati

Il modulo esporta i seguenti parametri:

1. `netm_ipv4`: l'indirizzo IP di ascolto del modulo NET-M;
2. `netm_port`: la porta TCP di ascolto del modulo NET-M;
3. `netm_resp_timeout`: massimo tempo di attesa per le risposte provenienti dal modulo NET-M, espresso in secondi (un valore inferiore ad uno indica un tempo infinito).

Esempio di impostazione dei parametri nel file di configurazione di OpenSIPS:

```
modparam("netres", "netm_ipv4", "127.1.2.3")
modparam("netres", "netm_port", 1846)
modparam("netres", "netm_resp_timeout", -1)
```

5.2.2 Funzioni esportate

Il modulo esporta le due seguenti funzioni:

1. `has_nrdl_request()`
2. `process_nrdl_request()`

5.2.3 Funzione `has_nrdl_request()`

Questa funzione consente di verificare se un messaggio SIP incapsula contenuto NRDL. In particolare, la funzione si accerta che la descrizione di risorse di rete contenuta nel messaggio rappresenti una richiesta di risorse di rete. Infatti solo le richieste di riservazione di risorse devono essere prese in considerazione e passate allo strato di trasporto.

Signature della funzione:

```
int has_nrdl_request(struct sip_msg *msg)
```

Valori di ritorno:

- `1` se il messaggio specificato contiene una richiesta NRDL;
- `-1` in caso contrario.

5.2.3.1 Implementazione

Dopo aver eseguito un veloce controllo sulla consistenza del parametro di ingresso, la funzione estrae dal corpo del messaggio la parte di tipo RDF/XML; la parte, cioè, che ha un valore di `Content-Type` pari ad `application/rdf+xml`[24]:

```
str rdf_part = {0, 0};
const str rdf_ctype = {RDF_CONTENT_TYPE_STR,
   RDF_CONTENT_TYPE_LEN};

multipart_get_content(msg, &rdf_ctype, &rdf_part);
```

Nel caso in cui il messaggio contenga una parte di RDF, ne viene eseguito il parsing utilizzando la libreria Raptor[3]:

```
init_raptor();

raptor_set_statement_handler(rdf_parser, NULL, parse_triple
   );
raptor_set_namespace_handler(rdf_parser, NULL,
   namespace_watcher);

r = raptor_start_parse(rdf_parser, base_uri);
if (r != 0)
   goto stop_raptor;

raptor_parse_chunk(rdf_parser, (void *)rdf_part.s, rdf_part
   .len, 1);
```

Finito il parsing, la funzione ritorna il valore affermativo (1) se e solo se il contenuto RDF contiene la dichiarazione del namespace NRDL e nel documento non vi sono classi con un parametro `answer`:

```
stop_raptor:
   fini_raptor();

end:
   if (has_namespace && !has_answer)
      return 1;
   else
      return -1;
```

5.2.4 Funzione `process_nrdl_request()`

Questa funzione esegue le seguenti operazioni:

- estrae la richiesta di riservazione di risorse di rete (NRDL) dal corpo di un messaggio;

[3]Raptor: RDF Parser Library, `http://librdf.org/raptor/`

- trasmette la richiesta allo strato di trasporto della rete NGN, cioè al software che implementa l'interfaccia R_S;
- attende la risposta (sempre in formato NRDL) proveniente dallo strato di trasporto;
- sostituisce, nel corpo del messaggio, la richiesta con la risposta appena ricevuta dallo strato di trasporto.

Signature della funzione:

```
int process_nrdl_request(struct sip_msg *msg)
```

Valori di ritorno:

- `1` se il messaggio può essere inoltrato;
- numero negativo se il messaggio non deve essere inoltrato.

5.2.4.1 Implementazione

Prima di tutto, viene creata una lista di tutte le parti contenute all'interno del corpo del messaggio, dopodiché si trova la parte di interesse (quella RDF), facendo attenzione al fatto che ve ne sia solo una nel corpo del messaggio:

```
memset(&body_list, 0, sizeof(typed_body));
multipart_get_all_contents(msg, &body_list);

for (b = &body_list; b != NULL; b = b->next) {
   if (b->body.s == NULL || b->body.len <= 0)
      continue;

   nparts++;

   if (strncmp(b->content_type.s, RDF_CONTENT_TYPE_STR,
      RDF_CONTENT_TYPE_LEN) == 0) {

      if (request.len > 0) {
         r = ERR_TOO_MANY_RDFS;
         goto on_error;
```

```
        }

        request.s = b->body.s;
        request.len = b->body.len;
    }
}

if (request.len <= 0 || request.s == NULL) {
    r = ERR_INTERNAL;
    goto on_error;
}
```

Una volta estratto il contenuto RDF, lo si utilizza per la riservazione delle risorse di rete:

```
r = reservation_req_resp(&request, &response);
if (r != 0)
    goto on_error;
```

Viene eseguito un controllo sulla risposta. In particolare, vige la regola secondo cui se la risposta è uguale alla richiesta, allora la richiesta non è ben formata:

```
if (strncmp(request.s, response.s, request.len) == 0) {
    LM_ERR("NRDL request not accepted by Transport Stratum\n
        ");
    r = ERR_REQ_NOT_ACCEPT;
    goto on_error;
}
```

Occorre notare che la SCF (proxy SIP) non si interessa del contenuto della risposta, quindi si comporta ugualmente nel caso di risposta positiva o negativa. Infatti sarà compito dell'entità a cui è destinato il messaggio capire se la riservazione delle risorse di rete richieste dall'altra parte è andata a buon fine, e comportarsi di conseguenza.

Se la richiesta è stata interpretata con successo dallo strato di trasporto, allora si procede alla creazione di un nuovo messaggio che contenga la risposta appena ricevuta al posto della richiesta iniziale. Il nuovo messaggio sostituirà quello ricevuto come parametro di ingresso della funzione:

```
for (b = &body_list; b != NULL; b = b->next) {
```

```
    if (b->body.s == NULL || b->body.len <= 0)
        continue;

    if (strncmp(b->content_type.s, RDF_CONTENT_TYPE_STR,
        RDF_CONTENT_TYPE_LEN) == 0) {
        b->body.s = response.s;
        b->body.len = response.len;
        break;
    }
}

r = multipart_build_msg(msg, &body_list);
if (r != 0) {
    r = ERR_INTERNAL;
    goto on_error;
}
```

Se la creazione del messaggio va a buon fine, si dealloca la memoria precedentemente allocata da `multipart_get_all_contents()`, e si ritorna il valore `1`:

```
for (b = &body_list; b != NULL && b->next != NULL; b = b->
    next);
for (; b != NULL; b = b->prev)
    pkg_free(b->next);

LM_DBG("NRDL request successfully replaced with response\n"
    );
return 1;
```

Nei casi di errore visti in precedenza si salta alla label `on_error`. Qui per prima cosa si dealloca la memoria dinamica allocata in precedenza, quindi, a seconda del tipo di messaggio SIP che si sta modificando, l'esecuzione si differenzia. Se il messaggio è di richiesta si ritorna il valore di errore, negativo, che dipende dalla tipologia di errore riscontrato:

```
on_error:
for (b = &body_list; b != NULL && b->next != NULL; b = b->
    next);
for (; b != NULL; b = b->prev)
    pkg_free(b->next);
```

```
if (msg->first_line.type == SIP_REQUEST)
   return r;
```

Se invece il messaggio attualmente considerato è una risposta SIP (ad esempio 183), deve essere inoltrato in ogni caso perché se andasse perduto, il SIP User Agent che lo ha spedito (nel caso in cui venga usata l'estensione per l'affidabilità delle risposte provvisorie – metodo `PRACK`) continuerebbe ad inviarlo fino allo scadere di un apposito timeout. Per questo motivo, nel caso in questione, è stata fatta la scelta implementativa di sostituire la richiesta NRDL con un documento RDF non valido, in maniera che l'applicazione ricevente possa rendersi conto dell'occorrenza di un errore interno al Service Stratum NGN e comportarsi di conseguenza:

```
memset(&body_list, 0, sizeof(typed_body));
body_list.next = NULL;
body_list.prev = NULL;
body_list.content_type.s = RDF_CONTENT_TYPE_STR;
body_list.content_type.len = RDF_CONTENT_TYPE_LEN;
body_list.body.s = "ERRORED.";
body_list.body.len = 8;

r = multipart_build_msg(msg, &body_list);
if (r != 0) {
   LM_ERR("this response will be lost forever...\n");
   return -1;
}

return 1;
```

5.2.5 Altre funzioni

Di seguito sono elencate le funzioni di maggior interesse che sono state implementate all'interno del modulo Netres.

- `multipart_get_all_contents()`: estrae il corpo (body) di un messaggio, o tutte le sue parti se è in formato multipart.

- `multipart_get_content()`: estrae dal corpo di un messaggio solo la parte avente un determinato `Content-Type`.
- `multipart_build_msg()`: crea un nuovo messaggio SIP (in formato multipart se necessario) a partire da un messaggio esistente, mantenendone le parti essenziali dell'intestazione e creando il corpo in base ad una lista di contenuti, ognuno con il proprio `Content-Type`.
- `reservation_req_resp()`: si connette tramite socket TCP al modulo NET-M, trasmette la richiesta di riservazione di risorse di rete, e infine attende la risposta per un tempo che può essere superiormente limitato (a seconda del valore assegnato a `netm_resp_timeout`).

5.2.6 Script di routing

Per conferire all'applicazione il comportamento desiderato è necessario configurare tale comportamento all'interno del file di configurazione di OpenSIPS; in particolare si devono mettere insieme, nella maniera appropriata, le funzioni esportate dai moduli caricati. Come già specificato, i luoghi in cui vengono impostati i comportamenti sono i cosiddetti script di instradamento dei messaggi.

Esistono diverse famiglie di script, le due principali sono quella associata ai messaggi di richiesta (`route`) e quella associati ai messaggi di risposta (`onreply_route`). Per ognuna delle due famiglie è possibile specificare diversi script (es. `route[0]`, `route[1]`, `route[2]`, ecc.); lo script eseguito direttamente da OpenSIPS è quello con l'indice più basso, quindi, ad esempio, se fossero definiti i seguenti script di risposta: `onreply_route` (l'indice zero può essere sottinteso), `onreply_route[1]` e `onreply_route[2]`, all'arrivo di un messaggio di risposta l'applicazione eseguirebbe lo script `onreply_route`. Dall'interno di questo, poi, se ne potranno richiamare altri, ma solo in maniera mutuamente esclusiva, infatti una volta raggiunta la fine di uno script, il trattamento del messaggio corrente termina e viene preso in considerazione quello successivo.

Gli script di routing di OpenSIPS utilizzano una sintassi molto simile a quella del linguaggio di programmazione C.

Le funzionalità offerte dal modulo Netres sono state usata all'interno degli script `route` e `onreply_route`, per poter intercettare sia i messaggi di richiesta che quelli di risposta.

5.2.6.1 Script `route`

Come sottolineato in precedenza, lo script `route` viene eseguito ogni qual volta OpenSIPS riceve un messaggio SIP di richiesta.

Per comodità si è deciso di considerare solo i messaggi di richiesta `INVITE` nella ricerca di contenuto NRDL. Essendo messaggi di richiesta, è possibile gestire in maniera diretta le situazioni di errore, quindi quando queste si presentano il proxy risponde al SIP User Agent mittente del messaggio in maniera "stateless" e coerentemente con il valore di ritorno della funzione `process_nrdl_request()`, cioè con l'errore riscontrato durante il tentativo di riservazione delle risorse di rete.

Il codice seguente è stato inserito all'inizio dello script `route`:

```
if (is_method("INVITE") && has_nrdl_request()) {
  sl_send_reply("100", "Trying");
  if (!process_nrdl_request()) {
    switch ($retcode) {
      case -1:
        sl_send_reply("500", "Internal error");
        exit;
      case -2:
        sl_send_reply("503", "Resource reservation service
           not available");
        exit;
      case -3:
        sl_send_reply("504", "Resource reservation timeout
           ");
        exit;
      case -4:
        sl_send_reply("500", "Resource reservation response
            too large");
        exit;
      case -5:
        sl_send_reply("406", "Resource reservation request
           not accepted");
```

```
        exit;
      case -6:
        sl_send_reply("485", "Ambiguous (too many RDFs)");
        exit;
    }
  }
}
```

Quando l'esecuzione di `process_nrdl_request()` ritorna un valore positivo, significa che il contenuto NRDL nel corpo del messaggio è stato sostituito con la risposta di riservazione delle risorse di rete. In questo caso il messaggio subisce la sorte predefinita dei messaggi che giungono ad OpenSIPS: viene reinstradato, possibilmente al destinatario del messaggio, oppure ad un altro proxy più "vicino" (in termini retistici) alla destinazione, in base alle logiche definite dal protocollo SIP.

5.2.6.2 Script `onreply_route`

Questo script è eseguito quando giungono ad OpenSIPS messaggi di risposta.

Per convenzione, le risposte considerate sono solo quelle aventi codice di stato pari a `183`, cioè quelle che assumono il significato standard in SIP "Session Progress"[3]. Nel caso di messaggi di risposta non c'è la possibilità per il proxy di inviare una risposta allo User Agent mittente; l'unica cosa che esso può fare è inoltrare il messaggio comunque, come già specificato. Lo script `onreply_route` è stato creato ex-novo, in quanto non presente nel file di configurazione di default. Si noti che in questo caso la funzione `process_nrdl_request()` ritorna un valore negativo solo se non riesce a costruire il messaggio da inoltrare, quindi solo in presenza di un grave stato di errore.

```
onreply_route {
  if (status == "183" && has_nrdl_request()) {
    if (!process_nrdl_request())
      exit;
  }
}
```

5.3 Modulo NET-M

L'interfaccia R_S nell'architettura di rete NGN è l'elemento funzionale che rende possibile la comunicazione fra la SCF del Service Stratum e le funzioni RACF del Transport Stratum.

Il modulo NET-M è, nell'architettura di rete proposta, l'entità adibita alla comunicazione con i DSE SOON (fig. 8). In particolare, NET-M invia al DSE una serie di richieste di servizio derivanti dall'opportuna conversione delle richieste ad alto livello di riservazione di risorse di rete (espresse secondo il vocabolario NRDL), fatte dalle applicazioni. Per quanto detto, e tenuto conto della mappatura fra rete SOON e Transport Stratum NGN indicata nel capitolo precedente, si può affermare che la comunicazione fra NET-M e DSE rappresenti di fatto l'interfaccia R_S dell'architettura NGN.

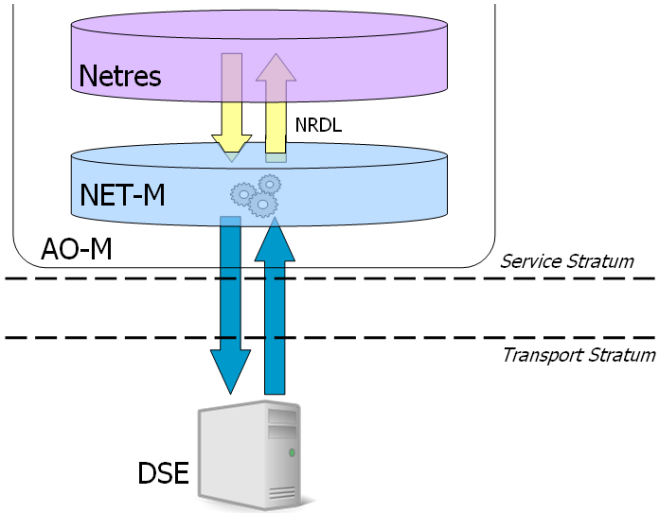

Figura 8. Modulo NET-M

Il software che implementa il modulo NET-M è stato ribattezzato "R2DAB – RDF to DSE and back" in virtù del suo compito, che è quello di convertire richieste NRDL, in formato RDF, in richieste di servizio per il DSE, e convertire le risposte del DSE in formato RDF.

La comunicazione con il DSE e con il modulo SIP-M avviene attraverso connessioni TCP.

5.3.1 Caratteristiche

R2DAB è in grado di accogliere e processare una sola richiesta per volta. La ragione è insita nel fatto che il DSE stesso funziona gestendo una richiesta alla volta, di conseguenza accodare le richieste pervenute durante una precedente elaborazione non avrebbe portato alcun beneficio in termini di throughput, ma solo un inutile aumento dell'occupazione di memoria, peggiorando le prestazioni complessive. Inoltre, un accodamento delle connessioni TCP avviene già a livello di sistema operativo.

R2DAB è in grado di interpretare richieste di riservazione di risorse di rete descritte secondo il vocabolario NRDL, all'interno di un documento RDF/XML. Inoltre, l'applicazione esegue la traduzione di tali richieste in una serie di richieste di servizio atomiche interpretabili dal DSE; in particolare, richieste di tipo Policy Enforcement.

Le comunicazioni con il DSE seguono un pattern semplice di richiesta-risposta; dopo ogni richiesta inviata, R2DAB attende la risposta dal DSE. Una volta ricevuta la risposta, il documento RDF/XML originale viene aggiornato con l'inserimento di opportuni parametri `answer`, indicanti lo stato di riservazione delle risorse.

R2DAB permette di specificare il valore dei parametri più salienti attraverso un file di configurazione testuale, il quale viene letto all'avvio dell'applicazione. È possibile specificare:

- indirizzo IP su cui ricevere connessioni in ingresso (richieste NRDL);
- porta TCP su cui ricevere connessioni in ingresso;
- indirizzo IP del DSE;
- porta TCP del DSE;
- timeout (in secondi) di ricezione per le richieste NRDL, una volta stabilita una connessione in ingresso;
- timeout (in secondi) di ricezione per le risposte del DSE.

Un esempio di file di configurazione:

```
server_addr = 127.1.2.3
server_port = 1846
```

```
dse_addr    = 10.0.0.10
dse_port    = 4195
rdf_timeout = 2
dse_timeout = -1
```

5.3.2 Implementazione

R2DAB è un'applicazione implementata interamente in C, sviluppata su un sistema GNU/Linux per essere eseguita su sistemi GNU/Linux, tuttavia è stata progettata in maniera da rendere possibile una non difficile esportazione verso piattaforme diverse (es. MS Windows). Essa si divide in una serie di file sorgente, ognuno dei quali apporta funzionalità distinte:

- `r2dab_config.c`: implementa le funzioni necessarie all'uso di un file di configurazione esterno (lettura del file per l'estrapolazione dei valori dei parametri);
- `r2dab_dse.c`: implementa le funzioni che consentono di mappare le descrizioni di risorse di rete in richieste per il DSE, nonché le funzionalità atte all'interpretazione delle risposte provenienti dal DSE.
- `r2dab_time.c`: mette a disposizione le funzionalità associate all'attesa temporale e alla rilevazione del tempo in genere;
- `r2dab_net.c`: implementa le funzioni di connessione, disconnessione e ricetrasmissione di dati attraverso socket TCP;
- `r2dab_rdf.c`: implementa le funzioni di parsing e serializzazione RDF/XML (grazie alla libreria Raptor), utilizzate nella fase di ricezione delle richieste e nella fase di aggiornamento del documento;
- `r2dab_fsm.c`: implementa la macchina a stati finiti dell'applicazione;
- `r2dab_main.c`: implementa la funzione `main()` ed è usato per inizializzare l'applicazione (con la lettura del file di configurazione) e avviare la macchina a stati finiti.

5.3.2.1 Macchina a stati finiti

La macchina a stati finiti di R2DAB (fig. 9) è implementata per intero all'interno della funzione `fsm_start()`, nel file `r2dab_fsm.c`.

Signature della funzione:

```
void fsm_start(void)
```

La funzione è suddivisa in due macroblocchi: il blocco iniziale – di inizializzazione – e il ciclo infinito.

Il blocco di inizializzazione viene eseguito solo una volta, alla chiamata della funzione. In questo blocco vengono impostate le configurazioni relative all'indirizzo del DSE e vengono create due socket TCP: la prima è quella usata per le richieste NRDL, mentre la seconda è quella usata per la comunicazione con il DSE:

```
memset(&dse_address, 0, SOCKADDR_SIZE);
if (inet_aton(cfg.dse_addr, &dse_address.sin_addr) != 1) {
   perror("inet_aton()");
   exit(1);
}

dse_address.sin_family = AF_INET;
dse_address.sin_port = htons(cfg.dse_port);

nrdl_sock = net_new_tcp_listen_socket(cfg.srv_addr, cfg.
   srv_port, 1);
if (nrdl_sock < 0) {
   printf("[%s] Error creating listening TCP socket\n",
      THIS_FILE);
   exit(1);
}

dse_sock = net_new_tcp_socket();
if (dse_sock < 0) {
   printf("[%s] Error creating TCP socket\n", THIS_FILE);
   exit(1);
}
```

Il ciclo infinito inizia con l'attesa di una connessione in ingresso. Nonappena il modulo Netres stabilisce la connessione con R2DAB, quest'ultimo si appresta a ricevere la richiesta NRDL:

```
while (1) {
   switch (state) {
      case WAIT_REQUEST:
         requestor_address = (struct sockaddr_in){0};
         l = SOCKADDR_SIZE;
         memset(&requestor_address, 0, SOCKADDR_SIZE);
         printf("\n[%s] Accepting incoming connection\n",
            THIS_FILE);

         requestor_sock = accept(nrdl_sock, (struct
            sockaddr *)&requestor_address, &l);
         status = net_tcp_rx(&requestor_sock, rdfbuf,
            BUF_SIZE, &requestor_msg, cfg.rdf_timeout);
         switch (status) {
            case STATUS_OK:
               state = REQUEST_RX;
               break;
            case STATUS_TIMEOUT:
               printf("[%s] RDF timeout expired (%is)\n",
                  THIS_FILE, cfg.rdf_timeout);
               state = RESET_STATE;
               break;
            case STATUS_NO_DATA:
               printf("[%s] Received zero-byte message\n",
                  THIS_FILE);
               state = RESET_STATE;
               break;
            default:
               state = RESET_STATE;
   }
   break;
```

Una volta ricevuta la richiesta, si ha una transizione di stato. Nello stato successivo viene eseguito il parsing della richiesta, estratte le informazioni di connessione e viene apposta a tutte le risorse di rete richieste la risposta iniziale standard "Progress", la quale indica semplicemente che la richiesta è stata ben interpretata. Inoltre viene con-

trollata la validità delle richieste rispetto ai parametri interpretabili dal DSE e viene estratto l'identificativo del client che ha fatto richiesta delle risorse di rete:

```
case REQUEST_RX:
   sprintf(session_id, "%x", (unsigned int)
      time_get_timestamp());
   printf("\n[%s] Serving request \"%s\" (%i bytes)\n",
      THIS_FILE, session_id, requestor_msg.slen);
   memset(&connections, 0, sizeof(rdf_connection_t));
   rdf_parse_string(&requestor_msg);
   rdf_get_connections(&connections);

   ncmd = 0;
   current_conn = &connections;
   while ((current_conn = current_conn->next) != NULL) {
      ncmd += 1;
      rdf_set_answer(current_conn->bandwidth, ANS_PROGRESS,
          NULL);
      rdf_set_answer(current_conn->traffic_class,
         ANS_PROGRESS, NULL);
   }

   if (ncmd == 0) {
      printf("[%s] Nothing to do with this request\n",
         THIS_FILE);
      state = RESET_STATE;
      status = net_tcp_tx(&requestor_sock, &requestor_msg);
      if (status != STATUS_OK)
      printf("[%s] Error: could not send request back to
         requestor\n", THIS_FILE);
      break;
   }

   status = dse_check_connections(&connections);
   if (status != STATUS_OK) {
      state = RESET_STATE;
      rdf_update_document();
      status = net_tcp_tx(&requestor_sock, &rdf_document);
      if (status != STATUS_OK)
```

```
            printf("[%s] Error: could not send response to
               requestor\n", THIS_FILE);
         break;
      }

      current_conn = connections.next;
      rdf_get_sip_client(&sip_client);
      if (sip_client.uri == NULL) {
         state = RESET_STATE;
         printf("[%s] Error: could not get SIP UA info\n",
            THIS_FILE);
         break;
      }

      printf("[%s] SIP UA ID: %s; name: %s\n", THIS_FILE, (
         char *)sip_client.uri, sip_client.id ? (char *)
         sip_client.id : "none");
      rdf_update_document();
      state = TX_COMMAND;
      break;
```

Terminata con successo l'elaborazione della richiesta, la macchina a stati finiti subisce un cambiamento di stato. Nello stato successivo avviene la creazione e la trasmissione di un singolo comando XML al DSE:

```
case TX_COMMAND:
   if (ncmd < 1) {
      state = RESET_STATE;
      break;
   }
   status = dse_set_command(inet_ntoa(requestor_address.
      sin_addr), session_id, current_conn);
   if (status != STATUS_OK) {
      printf("[%s] Error creating DSE command - exiting\n",
          THIS_FILE);
      current_conn = &connections;
      while ((current_conn = current_conn->next) != NULL) {
         rdf_set_answer(current_conn->bandwidth, ANS_ERROR,
             "application error");
```

```
         rdf_set_answer(current_conn->traffic_class,
            ANS_ERROR, "application error");
      }
      rdf_update_document();
      net_tcp_tx(&requestor_sock, &rdf_document);
      exit(1);
   }

tx_dse_cmd:
   status = net_tcp_tx(&dse_sock, &dse_command);
   if (status != STATUS_OK) {
      net_flush_socket(&dse_sock, xmlbuf, BUF_SIZE);
      net_close_socket(&dse_sock);
      status = net_tcp_connect(&dse_sock, &dse_address,
         dse_address_length);
      if (status == STATUS_OK)
         goto tx_dse_cmd;

      printf("[%s] Error sending command to DSE (%s:%i)\n",
          THIS_FILE, inet_ntoa(dse_address.sin_addr), ntohs
         (dse_address.sin_port));
      current_conn = &connections;
      while ((current_conn = current_conn->next) != NULL) {
         rdf_set_answer(current_conn->bandwidth, ANS_ERROR,
             "DSE not available");
         rdf_set_answer(current_conn->traffic_class,
            ANS_ERROR, "DSE not available");
      }
      rdf_update_document();
      net_tcp_tx(&requestor_sock, &rdf_document);
      net_flush_socket(&dse_sock, xmlbuf, BUF_SIZE);
      net_close_socket(&dse_sock);
      state = RESET_STATE;
      break;
   } else {
      printf("[%s] Sent command #%i to DSE\n", THIS_FILE,
         ncmd);
      state = WAIT_RESPONSE;
      break;
   }
```

Dopo l'invio del comando al DSE si attende la sua risposta, con la possibilità di un timeout. In caso di timeout o ricezione errata, vengono impostate le risposte alle risorse di rete relative alla connessione attualmente richiesta. Viene anche gestito il caso di una temporanea perdita di connettività con il DSE:

```
case WAIT_RESPONSE:
   status = net_tcp_rx(&dse_sock, xmlbuf, BUF_SIZE, &
      dse_resp, cfg.dse_timeout);
   if (status == STATUS_TIMEOUT) {
      printf("[%s] DSE timeout expired (%is)\n", THIS_FILE,
          cfg.dse_timeout);
      rdf_set_answer(current_conn->bandwidth, ANS_TIMEOUT,
         "DSE timeout");
      rdf_set_answer(current_conn->traffic_class,
         ANS_TIMEOUT, "DSE timeout");
      rdf_update_document();
      net_tcp_tx(&requestor_sock, &rdf_document);
      net_flush_socket(&dse_sock, xmlbuf, BUF_SIZE);
      net_close_socket(&dse_sock);
      state = RESET_STATE;
      break;
   } else if (status == STATUS_NO_DATA) {
      net_flush_socket(&dse_sock, xmlbuf, BUF_SIZE);
      net_close_socket(&dse_sock);
      status = net_tcp_connect(&dse_sock, &dse_address,
         dse_address_length);
      if (status == STATUS_OK) {
         state = TX_COMMAND;
         goto tx_dse_cmd;
      }

      printf("[%s] Error receiving DSE response\n",
         THIS_FILE);
      current_conn = &connections;
      while ((current_conn = current_conn->next) != NULL) {
         rdf_set_answer(current_conn->bandwidth, ANS_ERROR,
             "DSE lost");
         rdf_set_answer(current_conn->traffic_class,
            ANS_ERROR, "DSE lost");
      }
```

```
        rdf_update_document();
        net_tcp_tx(&requestor_sock, &rdf_document);
        net_flush_socket(&dse_sock, xmlbuf, BUF_SIZE);
        net_close_socket(&dse_sock);
        state = RESET_STATE;
        break;
    }
    state = RESPONSE_RX;
    break;
```

Una volta ricevuta la risposta dal DSE, viene chiusa la connessione con esso, quindi viene eseguito il parsing della risposta, al fine di stabilire il tipo di risposta da apporre alle risorse di rete che fanno capo alla risposta appena ricevuta. In caso di risposta non affermativa, l'operazione di riservazione delle risorse viene sospesa e il documento RDF/XML viene immediatamente trasmesso al modulo Netres. Se invece la risposta del DSE è affermativa, se ci sono altri comandi da inviare si passa al comando successivo, altrimenti si avanza di stato:

```
case RESPONSE_RX:
    printf("[%s] Got DSE response\n", THIS_FILE);
    net_flush_socket(&dse_sock, xmlbuf, BUF_SIZE);
    net_close_socket(&dse_sock);
    status = dse_parse_response(&dse_resp);
    switch (status) {
        case STATUS_OK:
            rdf_set_answer(current_conn->bandwidth,
                ANS_ALLOWED, NULL);
            rdf_set_answer(current_conn->traffic_class,
                ANS_ALLOWED, NULL);
            break;
        case STATUS_DSE_BW_NOTAVAIL:
            rdf_set_answer(current_conn->bandwidth,
                ANS_NOT_ALLOWED, "not available");
            rdf_set_answer(current_conn->traffic_class,
                ANS_NOT_ALLOWED, "not available");
            state = RESET_STATE;
            break;
        default:
```

```
            rdf_set_answer(current_conn->bandwidth, ANS_ERROR,
                "internal error");
            rdf_set_answer(current_conn->traffic_class,
               ANS_ERROR, "internal error");
            state = RESET_STATE;
      }
      rdf_update_document();
      free(dse_resp.ptr);

      if (state == RESET_STATE) {
         status = net_tcp_tx(&requestor_sock, &rdf_document);
         if (status != STATUS_OK)
            printf("[%s] Error: could not send response to
               requestor\n", THIS_FILE);
         break;
      }

      ncmd -= 1;
      current_conn = current_conn->next;
      state = ncmd ? TX_COMMAND : ALL_RESPONSES_RX;
      break;
```

Quando la riservazione delle risorse di rete è conclusa con successo, viene spedito al modulo Netres il documento RDF/XML, aggiornato con tutte le risposte; quindi viene ripristinato lo stato di attesa della macchina a stati finiti, liberata la memoria dinamica allocata durante l'esecuzione e distrutte tutte le strutture dati relative alla richiesta NRDL appena conclusa:

```
case ALL_RESPONSES_RX:
   status = net_tcp_tx(&requestor_sock, &rdf_document);
   if (status != STATUS_OK)
      printf("[%s] Error: could not send RDF to requestor\n
         ", THIS_FILE);
   printf("[%s] Request \"%s\" completed successfully\n",
      THIS_FILE, session_id);
   state = RESET_STATE;
   break;

case RESET_STATE:
   state = WAIT_REQUEST;
```

```
    net_flush_socket(&requestor_sock, rdfbuf, BUF_SIZE);
    net_close_socket(&requestor_sock);
    free(requestor_msg.ptr);
    ncmd = 0;
    rdf_destroy_data();
    memset(&sip_client, 0, sizeof(sip_client));
    break;
}
```

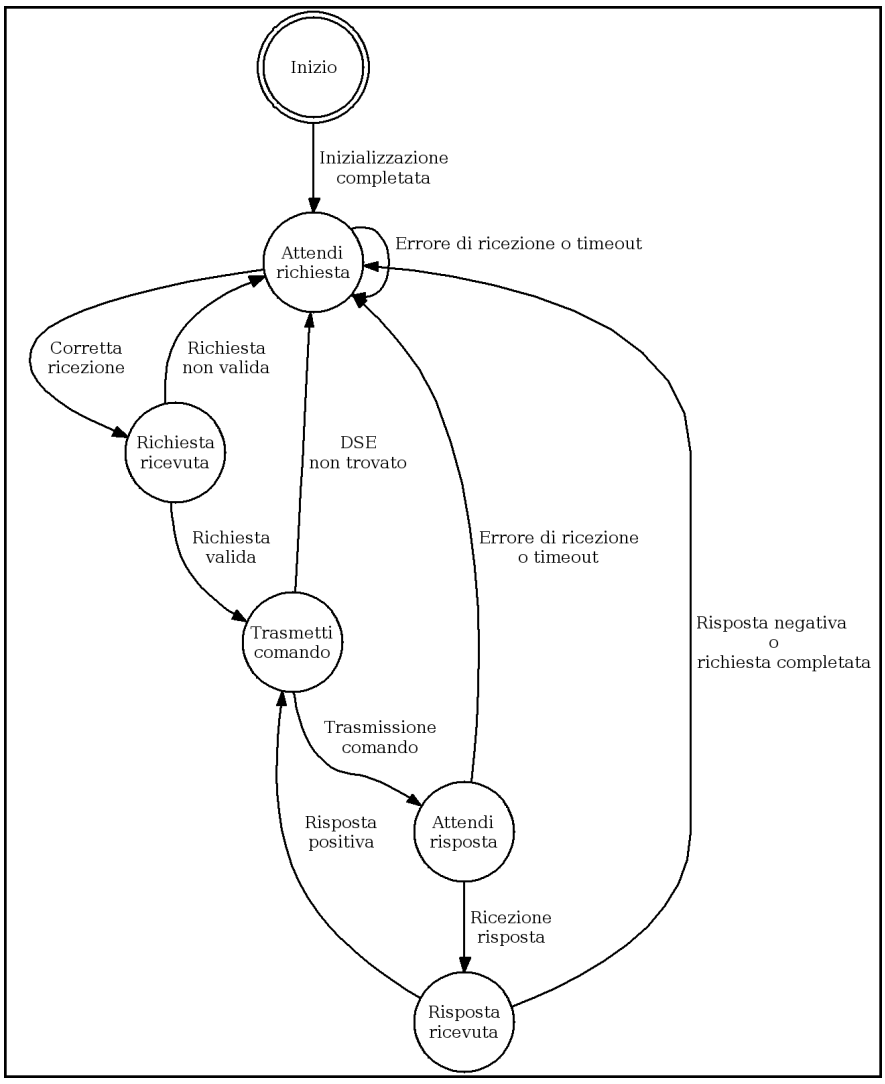

Figura 9. Macchina a stati finiti del modulo NET-M

5.4 SIP User Agent

Il software che implementa i SIP UA coinvolti nella segnalazione è stato chiamato "RESUA – RDF-enabled SIP User Agent". Questo software è interamente basato su PJSIP[4], che fornisce uno stack SIP lato client molto completo.

RESUA è stato sviluppato per essere un SIP UA semplice ma capace di inviare messaggi con body di tipo multipart. Esso ha due modalità di funzionamento. In modalità "client", può essere usato per iniziare una sessione SIP con un SIP UA remoto, trasmettendo un messaggio `INVITE` contenente sia la descrizione delle risorse di rete, che quella di risorse applicative. Invece in modalità "server" il software rimane in ascolto aspettando richieste `INVITE`; alla ricezione di un `INVITE`, esso risponde con un messaggio di risposta "183 Session Progress" contenente una descrizione di risorse di rete, infine risponde con "200 OK", stabilendo la sessione tra i SIP UA.

Questo software non implementa alcuna logica applicativa, i contenuti dei messaggi SIP vengono ricavati da opportuni file. Lo scopo di RESUA è unicamente quello di permettere il test dell'architettura proposta.

[4]PJSIP – Open Source SIP Stack, `http://www.pjsip.org/`

Capitolo 6

Validazione sperimentale

In questo capitolo viene introdotto il testbed sperimentale usato per la validazione dell'architettura proposta in questa tesi, quindi viene descritto l'esperimento condotto su tale testbed. L'esperimento ha il duplice obiettivo di verificare il funzionamento basilare dell'architettura e di valutare l'impatto della nuova segnalazione sui tempi di segnalazione complessivi.

6.1 Testbed

L'infrastruttura di prova è composta dai seguenti componenti hardware:

- **Service Plane SOON**: comprende 3 mini-computer equipaggiati con sistema operativo Linux (kernel 2.6) sui quali è eseguita un'istanza del software DSE. Il DSE_x controlla il router di bordo (edge router) ER_x (x = 1, 2, 3) attraverso direttive NETCONF[25]. Ogni mini-computer dispone di due interfacce Fast Ethernet, una viene utilizzata per la comunicazione con il router associato, mentre all'altra è assegnato un indirizzo pubblico per poter ricevere richieste di servizio da applicazioni esterne (modulo NET-M, in questo caso).
- **Rete di trasporto**: è composta da sei router IP/MPLS Juniper™ connessi fra loro come mostrato in figura 10. I router ER_1, ER_2 ed ER_3 sono configurati per essere router di confine, mentre i router CR sono configurati come router interni (core). Tutti e sei i router dispongono di interfacce Fast Ethernet e supportano le funzionalità MPLS, OSPF e RSVP.

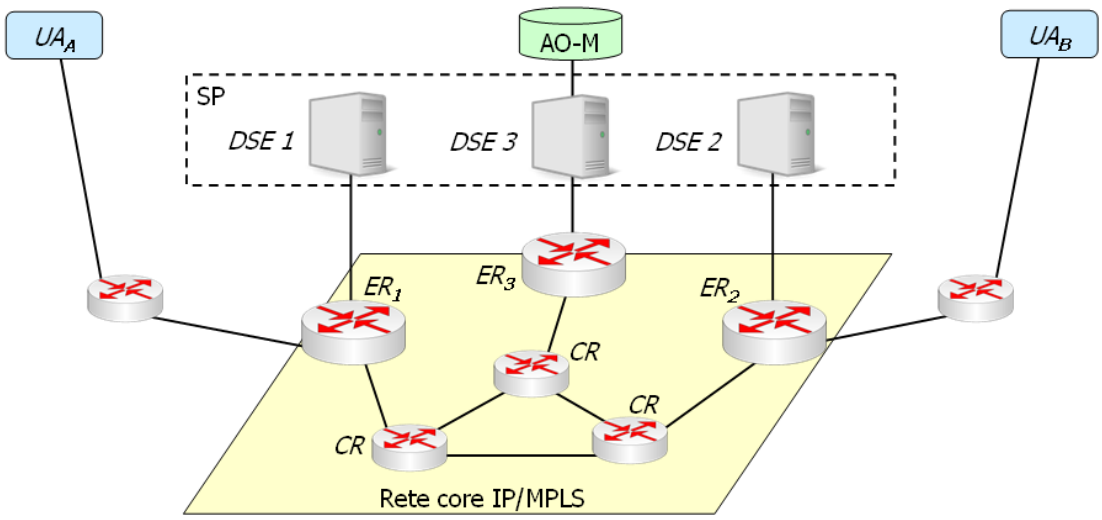

Figura 10. Testbed sperimentale

A DSE_3 è associato un modulo AO-M, il quale ricopre il ruolo di SCF e gestisce la segnalazione fra gli utenti, nonché la riservazione delle risorse di rete.

I SIP User Agent UA_A e UA_B rappresentano le applicazioni utenti della rete e sono due istanze del software RESUA, la prima in modalità "client" e la seconda in modalità "server". La segnalazione SIP tra i due UA non è diretta, ma passa attraverso il proxy determinato dal modulo SIP-M.

6.2 Test

Al fine di fornire una validazione dell'architettura proposta, è stato condotto un test di verifica sul testbed descritto in precedenza. L'esperimento è consistito nella creazione di una sessione di comunicazione tra le due applicazioni.

Nello scenario simulato, l'applicazione utente associata a UA_A intende fruire di un certo contenuto multimediale (es. un film in alta definizione) presente all'interno dello streaming server cui è associato UA_B. Per raggiungere il suo obiettivo l'applicazione utente, oltre a richiedere al server la risorsa applicativa, richiede al Service Stratum della rete, in maniera esplicita, le risorse di rete di cui necessita, con ben precisi parametri di qualità come bitrate e classe di traffico.

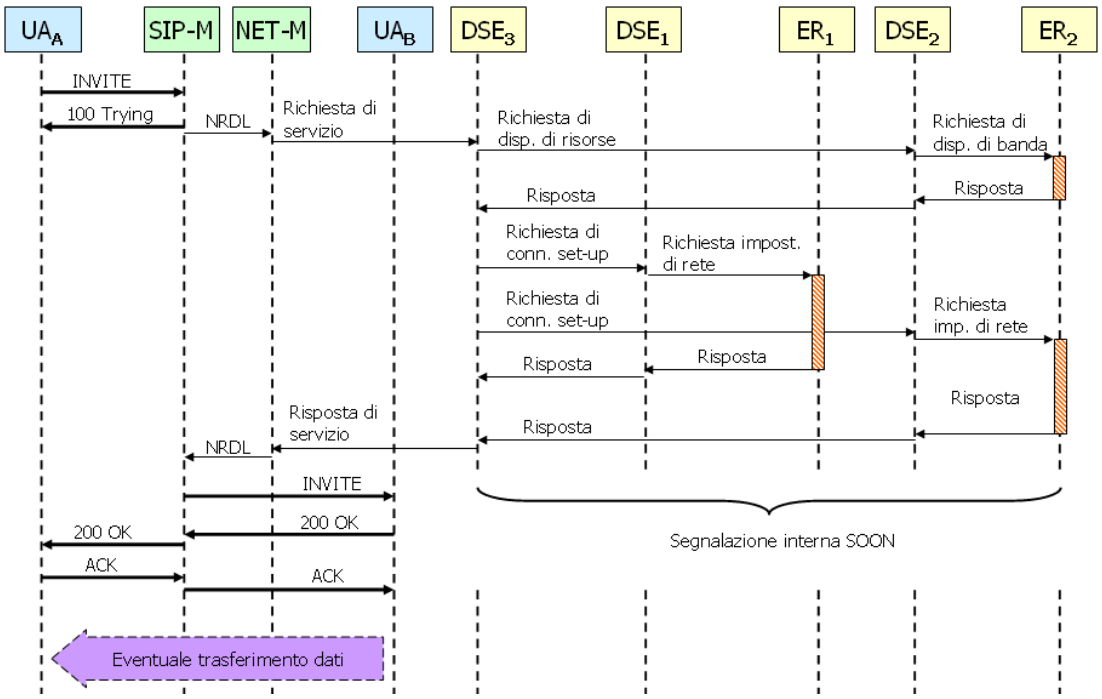

Figura 11. Scambio di messaggi durante il test

Il modulo SIP-M incorpora le funzionalità di proxy e registrar server, pertanto determina l'esistenza di un dominio SIP. In questo caso si assume che UA_A e UA_B siano stati precedentemente registrati e autenticati presso tale dominio.

6.2.1 Descrizione

L'esecuzione del test ha dato vita allo scambio di messaggi illustrato in figura 11, ricostruito grazie al software di cattura Wireshark[1]. Con riferimento a tale figura, UA_A dà inizio alla segnalazione trasmettendo al modulo SIP-M un messaggio SIP di richiesta di tipo `INVITE`. Questo messaggio è in formato multipart e al suo interno sono presenti due diversi contenuti: il descrittore delle risorse applicative e quello delle risorse di rete. La descrizione delle risorse applicative in questo caso contiene il titolo del film, l'indirizzo per la ricezione dei dati, i codec supportati, ecc.; tuttavia questo aspetto è al di là degli scopi di questa tesi, in quanto il funzionamento dell'architettura non ne è influenzato. La descrizione delle risorse di rete è in formato NRDL e rappresenta

[1] Wireshark network protocol analyzer, `http://www.wireshark.org/`

i requisiti di connettività dell'applicazione utente. Essa specifica la connessione che si intende stabilire e i relativi parametri di qualità (in questo caso bitrate e classe di traffico):

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:nrdl="http://deisnet.deis.unibo.it/nrdl/nrdl-
      syntax-ns#">

   <!-- Richiedente -->
   <nrdl:SIPDevice rdf:about="sip:client@example.net"/>

   <!-- Bitrate richiesta -->
   <nrdl:Bandwidth rdf:about="#bw">
      <nrdl:bandwidthValue>20000000</nrdl:bandwidthValue>
   </nrdl:Bandwidth>

   <!-- Classe di traffico richiesta -->
   <nrdl:TrafficClass rdf:about="#tc">
      <nrdl:trafficClassValue>Streaming</nrdl:
         trafficClassValue>
   </nrdl:TrafficClass>

   <!-- Indirizzo del server -->
   <nrdl:Endpoint rdf:about="#server">
      <nrdl:ipv4>10.20.10.2</nrdl:ipv4>
   </nrdl:Endpoint>

   <!-- Indirizzo del client -->
   <nrdl:Endpoint rdf:about="#client">
      <nrdl:ipv4>10.20.12.2</nrdl:ipv4>
   </nrdl:Endpoint>

   <!-- Connessione: i dati vanno dal server al client -->
   <nrdl:Connection rdf:about="#conn">
      <nrdl:source rdf:resource="#server"/>
      <nrdl:destination rdf:resource="#client"/>
      <nrdl:hasBandwidth rdf:resource="#bw"/>
      <nrdl:hasTrafficClass rdf:resource="#tc"/>
```

```
   </nrdl:Connection>

   <!-- Sessione comprendente un'unica connessione -->
   <nrdl:Session rdf:about="#session">
      <nrdl:name>Video Streaming session</nrdl:name>
      <nrdl:hasConnection rdf:resource="#conn"/>
   </nrdl:Session>
</rdf:RDF>
```

Il modulo SIP-M, in quanto proxy, trasmette ad UA_A la risposta temporanea "100 Trying", indicante che il messaggio è stato correttamente ricevuto e interpretato. Subito dopo, tale modulo estrae il contenuto NRDL dal messaggio e lo invia al modulo NET-M. Quest'ultimo trasmette la richiesta di servizio, debitamente derivata dal documento NRDL, a DSE_3.

A questo punto ha inizio la segnalazione interna alla rete SOON. Ricevuta la richiesta di servizio, DSE_3 risale agli edge router coinvolti nella comunicazione (ER_1 ed ER_2) e ai DSE che li controllano (DSE_1 e DSE_2), quindi verifica la disponibilità di risorse di rete (ovvero i 20 Mbit/s di banda richiesti) al router ER_2, in particolare, all'interfaccia verso ER_1, in quanto la richiesta di servizio riguarda i dati che da UA_2 fluiscono in direzione di UA_1. Questa verifica ha luogo attraverso una precisa richiesta di disponibilità di risorse a DSE_2. Una volta verificata la disponibilità di risorse, DSE_3 invia a DSE_1 e DSE_2 una richiesta di "connection set-up", ovvero di configurazione dei rispettivi router di bordo, al fine di stabilire la riservazione di una precisa quantità di banda (20 Mbit/s) e in una precisa classe di traffico, per i dati che vanno dall'indirizzo del server verso l'indirizzo del client.

Conclusasi la segnalazione interna del Service Plane SOON, il modulo NET-M riceve la risposta affermativa da DSE_3, che indica che la riservazione delle risorse di rete è andata a buon fine, così aggiorna il documento NRDL di richiesta con lo stato di riservazione delle risorse di rete (proprietà `answer`) e lo trasferisce al modulo SIP-M:

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
   ...
   <!-- Bitrate richiesta -->
   <nrdl:Bandwidth rdf:about="#bw">
      <nrdl:bandwidthValue>20000000</nrdl:bandwidthValue>
```

```
      <nrdl:answer>Allowed</nrdl:answer>
   </nrdl:Bandwidth>

   <!-- Classe di traffico richiesta -->
   <nrdl:TrafficClass rdf:about="#tc">
      <nrdl:trafficClassValue>Streaming</nrdl:
         trafficClassValue>
      <nrdl:answer>Allowed</nrdl:answer>
   </nrdl:TrafficClass>
   ...
</rdf:RDF>
```

Il modulo SIP-M sostituisce il documento NRDL iniziale con quello aggiornato e lo inoltra a UA_B. Questo risponde con un messaggio SIP "200 OK", infatti nella simulazione di video streamig il server non ha requisiti di connettività diversi da quelli che sono già soddisfatti per il client. Il messaggio di risposta viene inoltrato senza modifiche al client, il quale risponde con un messaggio di ACK, dando così vita alla sessione di comunicazione. In questo istante, in uno scenario reale, inizierebbe lo stream del contenuto multimediale.

6.2.2 Risultati

Dal test eseguito possono essere ricavati due risultati. Il primo è rappresentato dalla corretta creazione del canale di comunicazione richiesto fra i router ER_1 ed ER_2. Questo si evince dal file di configurazione di ER_2 dopo la riservazione delle risorse:

```
schedulers {
   EF-Scheduler {
      ...
   }
   AS-Scheduler {
      transmit-rate 20971520 exact;
      priority high;
   }
   BE-Scheduler {
      ...
   }
}
```

Quello mostrato è un estratto del file di configurazione di ER_2, in JUNOScript, relativo all'interfaccia di rete diretta verso la rete core. In esso è mostrata l'effettiva configurazione dello scheduler associato al traffico Assured Forwarding[26] ad alta priorità, per un tasso di trasmissione di circa 20 Mbit/s, a seguito della richiesta di servizio da parte del modulo NET-M. La velocità di trasmissione è in realtà impostata ad un valore di poco superiore a 20 Mbit/s per tollerare l'overhead introdotto all'interno della rete di trasporto.

Il secondo risultato riguarda il tempo relativo alla segnalazione extra introdotta dall'architettura proposta. A questo proposito il grafico di figura 12 rappresenta (non in scala) gli intervalli temporali più rilevanti durante la segnalazione. Si può osservare come il tempo di segnalazione interno SOON (in cui il tempo di configurazione dei router è largamente predominante) sia di un ordine di grandezza maggiore rispetto al ritardo introdotto dalla nuova segnalazione.

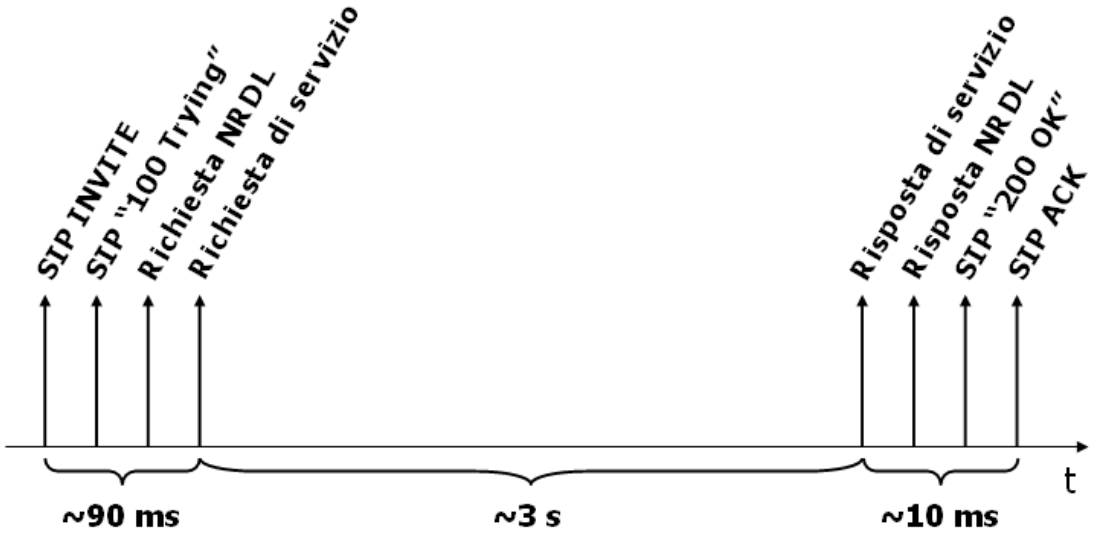

Figura 12. Durata temporale della segnalazione

Capitolo 7

Conclusioni

Questa tesi ha proposto un'architettura di rete basata sul concetto di application-aware networking. Alle applicazioni è data la possibilità di richiedere direttamente le risorse di rete di cui necessitano, tramite l'uso di un linguaggio generale e di alto livello che permette loro di astrarre completamente rispetto agli aspetti tecnologici e topologici relativi alle infrastrutture di accesso e di trasporto usate. Inoltre, lo sfruttamento del protocollo SIP dona all'architettura proposta una struttura di segnalazione robusta e meccanismi di supporto nelle fasi di pubblicazione e scoperta delle risorse di tipo applicativo.

È stato mostrato come l'architettura presentata in questo lavoro trovi corrispondenza con le linee guida ITU-T per le reti di nuova generazione (NGN).

Sono state presentate le implementazioni dell'alfabeto NRDL e dei moduli software fondamentali per una prima validazione dell'architettura da un punto di vista funzionale. Validazione che è stata eseguita sfruttando una particolare infrastruttura di rete, la rete SOON. Nonostante si riferiscano ad uno scenario estremamente limitato, i risultati del test permettono di trarre alcune conclusioni riguardo l'impatto dell'introduzione della nuova segnalazione all'interno di una infrastruttura di rete esistente. È possibile infatti affermare che le prestazioni globali, in termini temporali, restano per lo più invariate, in quanto il tempo di configurazione dei dispositivi di rete continua a rappresentare il maggior fattore di ritardo.

Alla luce di questo tipo di risultati è auspicabile un'estensione del lavoro svolto in questa tesi, che renda il moduo NET-M a sua volta modulare. Così facendo, l'architettura presentata potrà essere applicata a differenti reti di trasporto.

Appendice A

File di configurazione di OpenSIPS

```
#
# $Id: opensips.cfg 4423 2008-06-27 10:25:01Z henningw $
#

...

loadmodule "acc.so"

# ----- enable netres module -----
loadmodule "netres.so"

...

# ----- usrloc params -----
modparam("usrloc", "db_mode",   0)

# ----- netres params -----
modparam("netres", "netm_ipv4", "127.1.2.3")
modparam("netres", "netm_port", 1846)
modparam("netres", "netm_resp_timeout", -1)


####### Routing Logic ########


# main request routing logic
```

```
route {

  if (!mf_process_maxfwd_header("10")) {
    sl_send_reply("483","Too Many Hops");
    exit;
  }

  # ----- netres operation on INVITE -----
  if (is_method("INVITE") && has_nrdl_request()) {
    sl_send_reply("100", "Trying");

    if (!process_nrdl_request()) {
      switch ($retcode) {
        case -1:
          sl_send_reply("500", "Internal error");
          exit;
        case -2:
          sl_send_reply("503", "Resource reservation service
              not available");
          exit;
        case -3:
          sl_send_reply("504", "Resource reservation timeout
             ");
          exit;
        case -4:
          sl_send_reply("500", "Resource reservation
             response too large");
          exit;
        case -5:
          sl_send_reply("406", "Resource reservation request
              not accepted");
          exit;
        case -6:
          sl_send_reply("485", "Ambiguous (too many RDFs)");
          exit;
      }
    }
  }

...
```

```
  route(1);
}

onreply_route {
  # ----- netres operation on 183 -----
  if (status == "183" && has_nrdl_request()) {
    if (!process_nrdl_request()) {
      xlog("NRDL module: SIP reply lost\n");
      exit;
    }
  }
}

route[1] {

...
```

Bibliografia

[1] E. Mannie, "Generalized Multi-Protocol Label Switching (GMPLS) Architecture," RFC 3945, ottobre 2004.

[2] K. Knightson et al., "NGN Architecture: Generic Principles, Functional Architecture, and Implementation," *Communications Magazine*, IEEE, vol. 43, n. 10, ottobre 2005.

[3] J. Rosenberg et al., "SIP: Session Initiation Protocol," RFC 3261, giugno 2002.

[4] G. Zervas et al., "SIP-enabled Optical Burst Switching architectures and protocols for application-aware optical networks," *Computer Networks*, 2008, doi: 10.1016/j.comnet.2008.02.016.

[5] J. van der Ham, et al., "Using the Network Description Language in Optical Networks," 10th IFIP/IEEE International Symposium on Integrated Network Management (IM), maggio 2007.

[6] ITU-T Y.2001 "General Overview of NGN," dicembre 2004.

[7] H. Zimmermann, "OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection," *IEEE Transactions on Communications*, vol. COM-28, n. 4, pp. 425-432, aprile 1980.

[8] IETF, "Requirements for Internet Hosts – Communication Layers," RFC 1122, ottobre 1989.

[9] M. Handley et al., "SIP: Session Initiation Protocol," RFC 2543, marzo 1999.

[10] M. Handley et al., "SDP: Session Description Protocol," RFC 4566, luglio 2006.

[11] University of Southern California, "Transmission Control Protocol," RFC 793, settembre 1981.

[12] J. Rosenberg et al., "Reliability of Provisional Responses in the Session Initiation Protocol (SIP)," RFC 3262, giugno 2002.

[13] A. Niemi, "Session Initiation Protocol (SIP) Extension for Event State Publication," RFC 3903, ottobre 2004.

[14] A. B. Roach, "Session Initiation Protocol (SIP)-Specific Event Notification," RFC 3265, giugno 2002.

[15] B. Campbell et al., "Session Initiation Protocol (SIP) Extension for Instant Messaging," RFC 3428, dicembre 2002.

[16] O. Lassila, "Resource Description Framework (RDF) Model and Syntax Specification," W3C Recommendation, febbraio 1999, `http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/`

[17] G. Klyne et al., "Resource Description Framework (RDF): Concepts and Abstract Syntax," W3C Recommendation, febbraio 2004, `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`

[18] Resource Description Framework (RDF), `http://www.w3.org/RDF/`

[19] Extensible Markup Language (XML), `http://www.w3.org/TR/xml/`

[20] D. Brickley et al., "RDF Vocabulary Description Language 1.0: RDF Schema," W3C Recommendation, febbraio 2004, `http://www.w3.org/TR/rdf-schema/`

[21] T. Berners-Lee, "What the Semantic Web can represent," 1998, `http://www.w3.org/DesignIssues/RDFnot.html`

[22] B. Martini et al., "A Novel Service Oriented Framework for Automatically switched Transport Network," 9th IFIP/IEEE International Symposium on Integrated Network Management (IM), maggio 2005.

[23] F. Baroncelli et al., “Supporting Control Plane-enabled Transport Networks within ITU-T Next Generation Network (NGN) architecture,” Network Operation and Management Symposium (NOMS) 2008, Salvador, Brasile, aprile 2008.

[24] A. Swartz, “application/rdf+xml Media Type Registration,” RFC 3870, settembre 2004.

[25] R. Enns, “NETCONF Configuration Protocol,” RFC 4741, dicembre 2006.

[26] J. Heinanen et al., “Assured Forwarding PHB Group,” RFC 2597, giugno 1999.

Ringraziamenti

Devo un sentito ringraziamento al professor Franco Callegati, per avermi concesso la possibilità di affacciarmi su un mondo davvero stimolante – quello della ricerca. Inoltre devo ringraziarlo per la grande simpatia, disponibilità e pazienza adoperata nei miei riguardi in questi due anni di "Specialistica" e soprattutto durante il lavoro di tesi.

In secondo luogo è doveroso ringraziare il mio correlatore, Ing. Aldo Campi, per la sua attenta supervisione e la fiducia riposta nei miei confronti. Ringrazio inoltre il professor Walter Cerroni, che ha dato il suo grande contributo, materiale e morale, anche nella realizzazione di questa tesi.

Un ringraziamento speciale va a tutto il personale con sede a Pisa che in questi mesi mi ha messo a disposizione l'infrastruttura di rete SOON:

- Barbara Martini, Fabio Baroncelli (CNIT);
- Piero Castoldi, Karim Torkmen, Valerio Martini (Scuola Superiore Sant'Anna).

Tengo particolarmente a ringraziare Karim per le giornate intere passate con me su Skype™ supportandomi nel mio lavoro.

Un ringraziamento a Stefano Bianchini, Marco Ramilli e Nicola Calisesi, un trio di informatici che ha sicuramente cambiato la mia vita.

Ora è il momento di ringraziare coloro che hanno reso questi cinque anni di università un'esperienza unica, indimenticabile, ineguagliabile:

- Cola (Baggy), per la sua simpatia, intelligenza e autoironia;
- Vincenza, per non essersi ancora stancata di lui;

- Ivan (il bolgiaiolo), per sua grande capacità di resistere ai piccoli grandi scherzi cui la vita (nei panni miei e di qualcun'altro, vero Nico?) lo ha sottoposto in questi anni di università, ma soprattutto perché è e rimarrà per sempre un falso buonista;
- Valentina, per essere disponibile nel momento dello scherzo;
- Nico, per aver messo a disposizione la sua casa nelle cene universitarie, per aver messo i suoi ciliegi a disposizione mia e di Baggy, e per quell'aura da bravo ragazzo che lo circonda anche se in realtà nasconde una mente diabolica;
- Giova, per la sua inflessibilità e attitudine al comando;
- Maikol, per la sua pazzia sfrenata;
- Campe (F.d.F.), per le sue impareggiabili lezioni di vita e la sua creduloneria, che è seconda solo alla sua maestria con le donne;
- Jari (il finnico), per la sua grande generosità di spirito accompagnata da una malcelata acidità;
- Elisabetta, amica del finnico;
- Montini, per essere un provetto cacciatore di cinghiali e volatili, e per offrirci sublimi libagioni a casa sua;
- Ama, per averci spesso invitati a giocare e divertirci a casa sua;
- Moschino, perché è uno di noi e perché ci fece visitare luoghi stupendi vicino casa sua;
- Il Gagio, perché è un po' burbero, ma di cuore tenero;
- Crescia, che è semplicemente un genio;
- Simone Minardi, un motivo ci sarà;
- Migno, per avermi fatto provare i suoi pattini roller-blade una mattina in facoltà, per la sua incrollabile calma e per il suo salutismo portato all'estremo;

- Tama, per i suoi baffi caratteristici e la sua finta serietà;
- Brad, perché in fondo (ma molto in fondo) è un bravo ragazzo;
- Cisco, per essere la vera colonna portante della sede di Cesena della Seconda Facoltà di Ingegneria.

Sicuramente ho dimenticato qualcuno che avrei voluto includere. Mi perdoni!

Succintamente vorrei porgere un sincero ringraziamento a tutto il personale (docenti, segretari, tutor, dottorandi) di Ingegneria a Cesena, per il loro ammirevole impegno.

Voglio ringraziare, inoltre Nicoletta, Alice, Jessica, Daniele e Tommaso, per essermi stati vicini nell'ultimo anno di università.

Intendo ringraziare i miei genitori, Mario e Olimpia, per avermi sempre supportato e sostenuto, insieme alle mie due sorelle Ilaria e Lisa.

Inoltre è opportuno che io ringrazi Benny Prijono (autore di PJSIP), Dave Becket (autore di Raptor), Donald Knuth per aver inventato TEX, Leslie Lamport per averlo esteso creando LATEX, gli sviluppatori di LYX, per aver creato un così potente front-end per LATEX, infine gli sviluppatori di Linux e Debian, per aver costruito una piattaforma solida sulla quale svolgere la tesi.

Ringrazio, infine, Luca, Luca, Rainer, Michele, Michele e tutti i miei più vecchi amici che hanno sempre creduto in me. Ringrazio anche Martin e la sua famiglia, che da Frankenwinheim non hanno mai cessato di farmi sentire il loro incoraggiamento.

Ora è arrivato il momento di rendere grazie alla persona che, ancora una volta, ha realmente permesso che tutto questo si avverasse. Sto parlando della ragazza che mi sta vicino, sopportandomi, da oltre quattro anni. È generosa, intelligente, caparbia, onesta, profonda. Voglio ringraziarla perché è l'artefice della mie fortune e delle mie felicità. Ogni istante passato insieme a lei è la cosa più preziosa che esista. Se l'Amore avesse un nome sarebbe sicuramente *Elisa*.

Filippo,
Viserbella, 8 ottobre 2008